VŠB – Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

Katedra informatiky

2D počítačové zpracování fotografie na PC

2D computerized processing of photography on PC

2011 **Bc. Tomáš Pytlík**

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Dne 6. května 2011 Podpis: ……..

Rád bych touto formou poděkoval vedoucímu své diplomové práce panu doc. Ing. Lačezaru Ličevovi, CSc. za odborné vedení, trpělivost a ochotu v průběhu naší spolupráce.

Dále bych rád poděkoval své rodině a speciálně své sestře Marcele za pochopení a podporu v zajímavých časech.

Abstrakt

Ve své diplomové práci s názvem 2D počítačové zpracování fotografie se zabývám vytvořením nového modulu pro 2D modelování, který bude integrován do systému Fotom NG. Modul bude svým účelem a funkcionalitou navazovat na předchozí verzi modulu 2 aplikace Fotom 2008 avšak bude rozšířen tak, aby splňoval požadavky na moderní systém.

Modul bude implementován na platformě NetBeans.

V této práci se postupně zabývám studiem fotogrammetrie a 2D modelování, analýzou stávajících systému, specifikací požadavků pro nový modul, volbou softwarových prostředků a následným návrhem a implementací.

Součástí diplomové práce je také uživatelská příručka, programátorská dokumentace a dokument s popisem nutných změn ve Fotom API.

Klíčová slova:

fotogrammetrie, 2D modelování, Fotom NG, Fotom 2008,
NetBeans platforma, zpracování fotografie, zájmový objekt, měřický snímek

Abstract

In my diploma thesis named 2D computerized processing of photography on PC I am aiming to create new module for 2D modeling, which will be integrated into the Fotom NG system. Functionality and purpose of the module will be derived from previous version, which is module 2 of the application Fotom 2008, but it will be extended to meet requirements of a modern system.

The module will be solved as NetBeans platform based application.

This thesis is dedicated to study of photogrammetry and 2D modeling, analysis of current systems, new module requirements specification, software resources choice and following design and implementation.

Parts of thesis are also user's manual, programming documentation and document with description of necessary changes in Fotom API.

Keywords:

photogrammetry, 2D modelling, Fotom NG, Fotom 2008,
NetBeans platform, photography processing, object of interest, measurement image

Seznam použitých zkratek a symbolů

2D	Dvourozměrný
3D	Třírozměrný
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
CD	Compact Disc
CT	Computed Tomography
CSV	Comma Separated Values
EBT	Electron beam tomography
EPS	Encapsulated PostScript
GUI	Graphical User Interface
HTML	HyperText Markup Language
IDE	Integrated Development Environment
JDK	Java Development Kit
JPEG	Joint Photographic Experts Group (file format)
JRE	Java Runtime Environment
LGPL	GNU Lesser General Public License
MFC	Microsoft Foundation Classes
MRI	Magnetic resonance paging
OLE2	Object Linking and Embedding 2.0
OOXML	Office Open XML
PDF	Portable Document Format
PNG	Portable Network Graphics
RCP	Rich Client Platform
URL	Uniform Resource Locator
VŠB-TUO	Vysoká škola Báňská – technická univerzita Ostrava
XLS	Excel Spreadsheet File
XLSX	XML XLS
XML	Extensible Markup Language

Obsah

Seznam obrázků

Seznam tabulek

1. Úvod

Úkolem každého vědního oboru je kromě rozšiřování báze lidského vědění také usnadnění a vylepšení lidského života. Informatika, která patří mezi vědními obory k nejmladším, ale zároveň nejdynamičtěji se rozvíjejícím disciplínám, není výjimkou. Softwarové aplikace a systémy v dnešní době pomáhají člověku jak se zdánlivě obyčejnými činnostmi jako je účetnictví nebo vedení skladu, tak ve sférách životně důležitých jako je například lékařství. Jednou z takových aplikací je systém Fotom.

Myšlenka Fotomu vznikla v roce 1997 a jeho prvotním účelem bylo přeměřování důlních jam metodami fotogrammetrie. První verze programu spatřila světlo světa jako výsledek diplomové práce pana Ing. Holuši pod vedením pana doc. Ing. Lačezara Ličeva, CSc. V průběhu let se původně jednoduchý program vyvinul pod záštitou Fakulty elektrotechniky a informatiky VŠB-TUO v rozsáhlou aplikaci, která si ke své stávající funkčnosti přibrala ještě další cíl – stát se aplikací využitelnou v lékařství a medicíně k analýze snímků sond a neinvazivních vyšetření.

Z hlediska vývoje se stala přelomovou verze označená jako Fotom 2008. Aplikace obsahovala maximum funkcionality pro všemožné využití, avšak přinesla s sebou také mnoho nevýhod. Systém sestával z několika modulů, z nichž každý byl vyvíjen jako samostatná aplikace bez jakékoliv návaznosti na ostatní. Jediným pojítkem byl modul Fotom 1, který obsahoval tlačítka pro spuštění ostatních částí. Jak zmiňuje pan Ing. Lukáš Krahulec [1], neexistovalo žádné společné API nebo systémové jádro ani žádná metodologie vývoje. Programy pouze pracovaly nad jednotnou definicí vstupního souboru. Z tohoto pohledu se dala aplikace pouze velmi obtížně dále rozšiřovat. Dalším výrazným problémem byla absence programátorské dokumentace. Jakékoliv rozšíření nebo opravy chyb v aplikaci se tak stávaly velmi náročným úkolem. Moduly Fotomu 2008 byly napsány v programovacím jazyce C++ s využitím MFC a z technologického hlediska velmi rychle zastaraly.

Ještě v roce 2008 se ve vývoji Fotomu dospělo k rozhodnutí, že je třeba systém aktualizovat, sjednotit a odstranit jeho nevýhody, zvláště pak s důrazem na modularitu, budoucí rozšíření a nároky moderního softwaru. Systém byl označen jako Fotom 2009 a jednalo se o zcela novou aplikaci, která bude obsahovat již známou a praxí prověřenou funkcionalitu, a zhodnotí zkušenosti získané během vývoje předchozích verzí. V rámci své diplomové práce navrhl pan Ing. Lukáš Krahulec ve spolupráci s panem Ing. Janem Králem [2] nové jádro systému a na jeho základě pak zpracovali moduly pro definici objektů a kalibraci sondy.

Mým úkolem a obsahem této diplomové práce je vypracovat modul pro 2D modelování, který byl ve verzi 2008 označován jako Fotom 2. Integrací tohoto a několika dalších souběžně vyvíjených modulů vznikne nová generace systému Fotom označena jako Fotom NG[1].

Původní systém Fotom 2 nám dává dobrý náhled na základní funkcionalitu nového modulu. Z původního kódu ovšem nelze nic využít, protože Fotom 2009 je implementován ve velmi rozlišném prostředí a se zcela novým API. Bude proto zapotřebí nově specifikovat a analyzovat požadavky, vypracovat návrh modulu, provést implementaci a výsledné řešení otestovat. Průběh těchto úkonů postupně popisuje tento text.

V bezprostředně následující kapitole nejprve stručně pojednáme o principu a významu fotogrammetrie a 2D modelování. Další kapitola je věnována náhledu na Fotom 2008 a jeho následovníka Fotom 2009. Poté se zaměříme na již zmíněnou specifikaci požadavků včetně popisu použitých softwarových prostředků. Obsahem předposlední kapitoly je jejich analýza, návrh a implementace nového modulu. Práce je ukončena závěrem, ve kterém zhodnotíme všechny dosažené poznatky a výsledky.

[1] Tato zkratka pochází doslova z anglického sousloví *next generation*

2. Fotogrammetrie a 2D modelování

Protože se tato práce zaměřuje na počítačový systém sloužící ke zpracování fotogrammetrických snímků a konkrétním cílem práce je vytvořit modul pro 2D modelování, je vhodné nejprve blíže specifikovat obě tyto oblasti. V první části kapitoly se zaměříme na princip a význam fotogrammetrie. Druhá polovina je věnována způsobům 2D modelování. Níže uvedený text se opírá o informace uvedené v publikacích [3], [4] a [5].

2.1 Fotogrammetrie

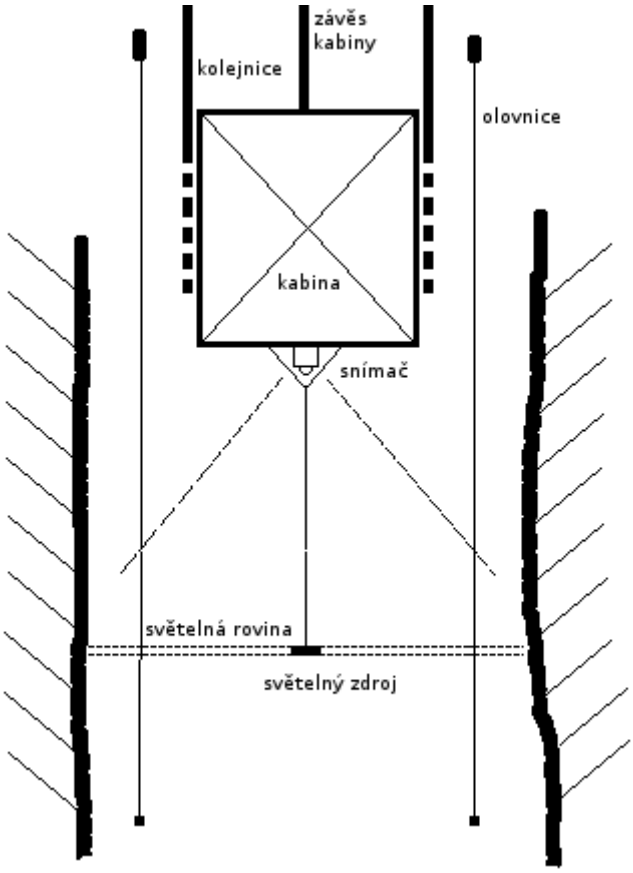
Fotogrammetrie je vědní obor, který se zaobírá určováním rozměrů, tvarů a polohy předmětů v prostoru podle fotografických snímků, pořízených z výšky nebo ze země, bez přímého proměřování. Základy této vědy byly položeny již během renesance a fotogrammetrie jako taková vznikla v roce 1851 v teoretické práci kartografa Aimé Laussedata. Navzdory faktu, že je často brána pouze jako odnož geodézie, jedná se o neustále se rozvíjející vědu, která má své nezastupitelné místo v mnoha oborech lidské činnosti.

Základní myšlenka fotogrammetrie vychází z principu jednoznačných geometrických vztahů mezi skutečným objektem reálného světa a jeho fotografickým snímkem. Velmi zjednodušeně řečeno se tato věda zaobírá určováním měřených hodnot skutečného předmětu dle jeho fotografie. Této metody se využívá tam, kde byl sběr dat reálného objektu velmi náročný nebo dokonce zhola nemožný (domovní fasády, mapování oblastí). Je samozřejmé, že zmíněná fotografie nemůže být ledajaká, nýbrž musí být vyfocena za speciálních podmínek a předem daným způsobem. V jazyce fotogrammetrie říkáme, že se jedná o tzv. *měřický snímek*. Tento snímek je za určitých podmínek (a nebereme-li v potaz určité fyzikální nesrovnalosti jako je zkreslení vlivem nedokonalého objektivu nebo snímkového materiálu) středovým průmětem dokumentovaného objektu. Využitím projektové geometrie lze ze získaných informací rekonstruovat tvar a rozměr předmětu.

Fotogrammetrii můžeme rozdělit do několika podkategorií podle různých hledisek. Například pořizujeme-li měřické snímky přímo v dané lokalitě přístrojem umístěným na stativu, pak se jedná o *pozemní* fotogrammetrii. Druhým způsobem získávání snímků je *letecká* fotogrammetrie, která slouží například pro potřeby kartografie.

Dalším kritériem, podle kterého můžeme fotogrammetrii rozdělit, je počet použitých snímků. Pokud je skutečnost zachycena pouze jediným snímkem, mluvíme o fotogrammetrii *jednosnímkové*. Na jejím základě můžeme měřit pouze rovinné vlastnosti objektu. Opakem tohoto případu je fotogrammetrie dvou- či obecně *vícesnímková*, která zachycuje tentýž objekt na několika snímcích, zpravidla ve více úhlech a tím nám dává možnost určit jeho pozici v prostoru a měřit prostorové vlastnosti sledovaného předmětu.

Fotogrammetrie ovšem nemusí sloužit pouze pro účely měření, ale také jako prostředek pro dokumentaci a sledování průběhu. Uveďme jako příklad snímkování důlní jámy. Fotografický přístroj je spouštěn do hloubky a pořizuje snímky v určitých vzdálenostech. Výsledkem je *série snímků* jednosnímkové fotogrammetrie, které jsou řazeny za sebou v daném pořadí a známých rozdílech vzdáleností. Ze získaných dat můžeme například určit, zda se profil důlní jámy s rostoucí hloubkou úží nebo rozšiřuje. Podobný příklad můžeme najít ve zdravotnictví, kdy lékař sleduje průběh kornatění tepen pacienta při onemocnění koronární aterosklerózou. Při této nemoci dochází k zužování průtoku krve tepnou vlivem obalování vnitřní strany tepny tukovými ložisky [6]. Lékař má k dispozici několik snímků získaných pomocí CT nebo EBT a jejich pořadí a pomyslné vzdálenosti jsou dány rozdílem času mezi snímkováním. Lékař tak může určit postup nemoci v čase.



Obrázek 1 - Snímkování důlní jámy [1]

Obrázek 2 - Lékařský snímek pořízený pomocí MRI

Existuje několik způsobů, jak zpracovávat měřický snímek. Analogové a analytické metody jsou v dnešní době vytlačeny do ústraní *digitální fotogrammetrií*. Na počátku analýzy již nestojí nepřesný obraz na skle nebo fotografickém papíře, ale jeho plně digitální obdoba ve vysokém rozlišení. Veškeré zpracování snímku se pak provádí automaticky ve vhodně navrženém softwaru. Příkladem budiž právě systém Fotom.

Fotogrammetrie se dnes využívá v mnoha vědeckých i průmyslových odvětvích. V předcházejícím textu již byly zmíněny příklady z kartografie, geodézie, stavebnictví, hornictví a zdravotnictví. Doplňme je pro zajímavost také o přínos fotogrammetrie v archeologii (pomocí leteckého snímkování se mapují rozsahy památek, odhalují pozůstatky starověkých sídlišť apod.), oceánografii (hlubinný výzkum a mapování pobřeží) nebo kriminalistice (vyšetřování dopravních nehod, odhalování důkazů na snímcích z místa činu).

2.2 2D modelování

Máme-li měřičský snímek v digitální podobě a chceme-li jej automatizovaně zpracovat na počítači, musíme nejprve snímek *interpretovat*. Jinými slovy je třeba přesně určit, které oblasti snímku zobrazují předměty, o které se zajímáme. Vzniklé objekty pak označujeme jako *zájmové*. Z hlediska počítačového zpracování je nanejvýš vhodné tyto objekty vyjádřit rovinnými matematickými útvary. Ty jsou přesně dané svými vzorci a je velmi snadné nad nimi provádět výpočty.

Rozeberme trochu více příklad důlní jámy uvedený v předcházející podkapitole. Máme k dispozici sérii snímků z různé hloubky. Na každém z nich označíme zájmový objekt – profil jámy – například pomocí kružnice. Víme, že snímky jdou za sebou v určité vzdálenosti. Na posledních snímcích (to jest z největší hloubky) se obsah kružnice zmenšuje. Z tohoto faktu můžeme usoudit, že se profil jámy se vzrůstající hloubkou zužuje. Jak ale zjistit přesný poměr zúžení? A jak můžeme odvodit obsah průřezu jámy v hloubce, ze které nemáme přesný snímek?

Získaná data můžeme vyjádřit v podobě grafu. Na ose x bude uveden námi sledovaný parametr - obsah průřezu jámy (kružnice). Osu y budou tvořit pořízené snímky v daném pořadí a s rozestupy odpovídajícími rozdílům v hloubce snímkování. V rovině grafu tak získáme několik bodů, které reprezentují naměřené hodnoty. Jestliže skrze ně interpolujeme křivku, získáme průběh zúžení profilu důlní jámy v závislosti na hloubce. Tento proces je jednou z několika metod 2D modelování, které budou popsány v dalším textu této práce.

2D modelování nám tedy slouží pro reprezentaci, porovnání, odečítání a sledování naměřených dat v průběhu měření.

3. Analýza stávajících systémů

Již v úvodu jsme předeslali, že byl Fotom 2008 rozdělen do škály modulů, kde každý zastával různorodou funkci. Například modul Fotom 1 se zabýval interpretací snímku, Fotom 3 dokázal vyobrazovat 3D reprezentaci modelované situace, Fotom 4 umožňoval 2D animaci snímku a tak dále. Středem našeho zájmu bude Fotom 2, jehož novou, revidovanou verzi budeme navrhovat.

V druhé části této kapitoly se pak zaměříme na Fotom 2009. Tato nová aplikace sice ctí základy svých předchůdců, ale je postavena na zcela nové platformě a vzhledu.

Protože nová generace programu Fotom NG bude obsahovat veškeré moduly Fotomu 2009 v nezměněné podobě, budeme pro jednoduchost o této aplikaci dále mluvit pouze jako o Fotomu NG.

3.1 Fotom 2

Druhý modul systému Fotom 2008 označovaný jako Fotom 2 byl určen pro 2D modelování. Základní kámen pro tento modul položil už pan Ing. Holuša v prvotní verzi Fotomu. Modul pak byl několikrát rozšiřován a přepracováván, například panem Ing. Koštuříkem v roce 2000 [8] a naposledy panem Ing. Žídkem v roce 2008 [7]. Tento vývoj sice velmi dobře vyprofiloval základní funkčnost a uživatelské prostředí aplikace avšak zároveň způsobil robustnost a nepřehlednost vnitřní implementace. Ta byla důvodem vzniku nových chyb, které již nebylo možno efektivně odstraňovat.

Modul nabízel několik různých výpočtů nad vstupními snímky, jejichž výsledky byly prezentovány ve formě grafů a tabulek. Mnoho z těchto výpočtů a postupů bude rovněž obsahem specifikace nového modulu, protože se jedná praxí ověřenou a uživateli oceněnou funkcionalitu.

Fotom 2 byl postaven na architektuře *document view*. Hlavní obrazovka byla krom menu tvořena zobrazovací plochou pro grafy a ovládacím panelem, který zprostředkovával přístup k funkcím programu.

Pokud chtěl uživatel provést výpočet, musel nejprve kliknout na tlačítko načtení snímků, které mu nabídlo příslušný dialog anebo dva po sobě jdoucí dialogy, pokud chtěl počítat nad dvěma sériemi snímků. Stejně tak nastavení vstupních dat výpočtu bylo prováděno přes dialogové okno a stejným způsobem se probíhalo zobrazení tabulky s výstupem. V jednu chvíli tak mohl mít uživatel na obrazovce až 4 samostatná okna. Běžná práce s modulem pak vyžadovala, aby uživatel takřka konstantně jednotlivé dialogy otevíral a uzavíral.

Vykreslování grafů bylo prováděno pomocí standardních možností jazyka C++. Přestože byla do systému implementována interpolace Beziérovou křivkou, byl výsledek často kostrbatý a vizuálně nevzhledný. Stejně tak GUI bylo postaveno na komponentách dostupných ve zmíněném prostředí, což mělo za následek například fakt, že je tabulka s výstupními daty reprezentována sérií listboxů.

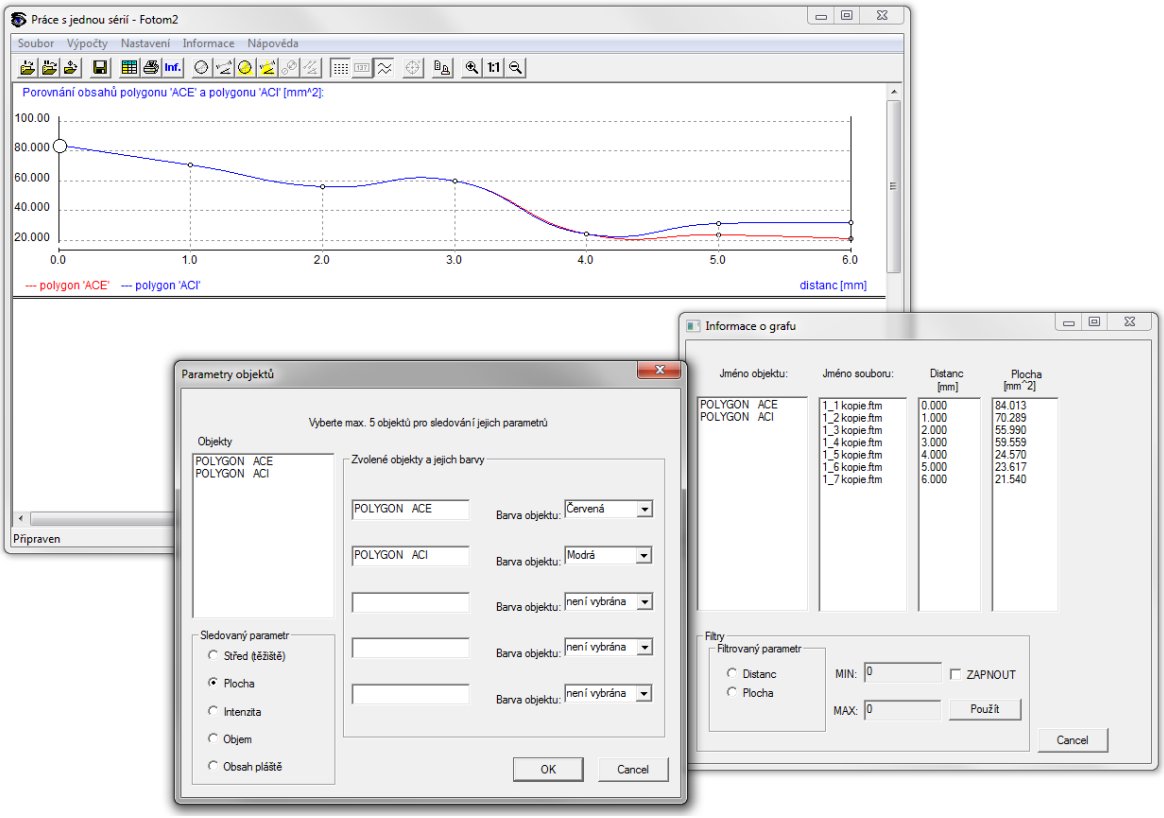
Modul umožňoval také export hodnot avšak pouze do textového souboru, kde byla uměle vytvořena tabulka pomocí ASCII znaků. Takovýto soubor byl prakticky dále nezpracovatelný.

Tyto a další nedostatky včetně různých chyb by měl nově navrhovaný modul odstranit.

3.2 Fotom NG

Vývoj soudobého systému Fotom NG začal v roce 2008 na základě aplikací Fotom 2008. U zrodu nové verze stáli krom vedoucího práce pana doc. Ing. Lačezara Ličeva, CSc. také pánové Ing. Lukáš Krahulec a Ing. Jan Král.

Hlavní myšlenkou nového vývoje bylo navrhnout jednotný moderní rozšiřitelný systém a především pak ucelené, zdokumentované a dobře použitelné API. S těmito prostředky pak výše



Obrázek 3 - Modul 2 systému Fotom 2008

zmínění pánové vytvořili jádro a prototyp budoucího programu, který zastával funkcionalitu původního Fotomu 1.

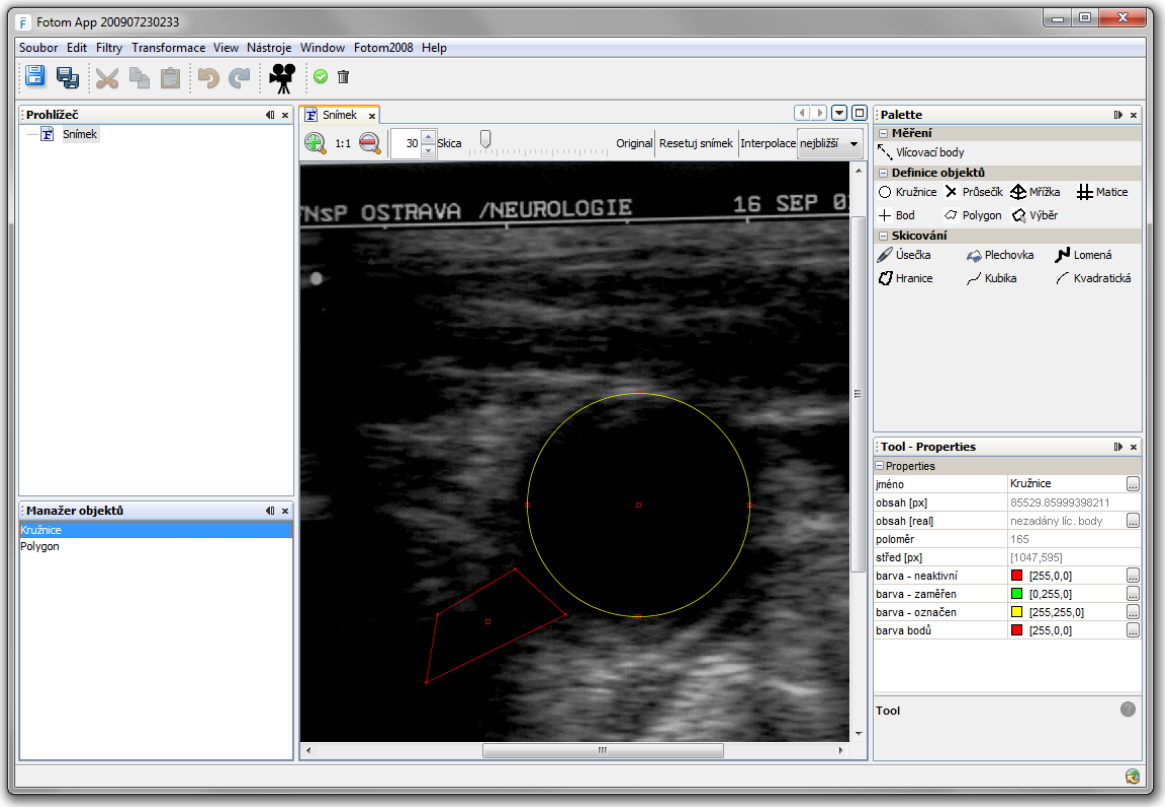
Pro implementaci byla zvolena platforma NetBeans, která umožňuje modulární vývoj softwaru, nabízí velký rozsah připravených komponent a navíc využívá všech možností platformy Java, jako jsou například knihovny komponent Swing.

Výsledný systém obsahuje integrované uživatelské rozhraní. Je postaveno na architektuře oken, která umožňuje uživateli pracovat buďto v režimu jednotného vzhledu anebo ve vlastním nastavení. Platforma se stará o to, aby bylo toto nastavení uloženo a při novém spuštění opět aplikováno. Okna jsou navíc dokovatelná, což znamená, že mohou být kdykoliv integrována do rozhraní anebo zobrazena jako samostatná.

Defaultní nastavení dělí obrazovku na pět částí. Vlevo je umístěn Průzkumník, který umožňuje vytvářet, zpravovat a otevírat jednotlivé snímky. Pod ním je okno Manažera objektů. Tato jednoduchá, leč velmi efektivní funkce zobrazuje zájmové objekty na snímku a umožňuje mezi nimi přepínat. Centrální část aplikace je věnována hlavnímu účelu systému – práci se snímkem. Jednotlivé snímky se otevírají jako záložky a každá z nich je opatřena vlastním ovládacím panelem pro rychlý přístup k nejběžnějším volbám. V levé části je zobrazena Paleta nástrojů pro interpretaci objektů na snímku a pod ní okno pro editaci vlastností.

Globální funkce a tlačítko pro zobrazení Manažera sond je umístěno v hlavním ovládacím panelu.

Krom uživatelského rozhraní a volby platformy je hlavní přínosem z hlediska vývoje nové Fotom API. Jeho analýze z pohledu návrhu modulu pro 2D modelování je věnována kapitola 5.2.2.



Obrázek 4 - Fotom NG

4. Specifikace požadavků na nový modul

Obsahem této kapitoly je přesné vyjádření všech požadavků a funkcionality, které má nový modul pro 2D modelování splňovat.

Požadavky vycházejí z funkcionality Fotomu 2008 a jsou rozšířeny dle praktických zkušeností získaných používáním tohoto programu a systému Fotom NG.

4.1 Definice pojmů

Abychom mohli požadavky specifikovat v jednoznačné a ucelené podobě je nejprve potřeba definovat pojmy, které k tomuto budeme využívat. Dříve uvedené intuitivní pojmenování zpřesníme, popřípadě vymezíme pro oblast popisovaného problému s přihlédnutím na jejich reprezentaci v systému Fotom NG.

4.1.1 Snímek

Slovem snímek budeme dále označovat měřický snímek v jeho digitální podobě. Téměř výhradně pak budeme mít na mysli datové soubory zpracované systémem Fotom NG.

Každý takový snímek v sobě obsahuje definice zájmových objektů, souřadnice jejich zájmových bodů, lícovací body a jejich skutečné hodnoty definici měřících jednotek a jednotek distance. Co je to distanc, lícovací body a jak se určují globální a lokální jednotky bude vysvětleno v podkapitolách 4.1.3 a 4.1.4.

Obecně je možné, aby datový snímek existoval bez definovaných lícovacích bodů, popřípadě bez nadefinovaných parametrů pro převod na globální jednotky. Takovýto snímek však nemá žádný matematicky daný vztah ke skutečnosti a proto je pro 2D modelování nepoužitelný. Budeme tedy předpokládat, že *snímek vhodný k 2D modelování*, je takový snímek, který lícovací body i parametry pro převod obsahuje.

Podobnou úvahou můžeme dospět k názoru, že nemá velký smysl provádět 2D modelaci na snímku, který nemá ani jeden zájmový objekt vhodný pro 2D modelování. Více bude o zájmových bodech vysvětleno v podkapitole 4.1.5.

Připomeňme také, že každý snímek má svůj název, kterým jej lze plně identifikovat.

4.1.2 Měření

Měření neboli *série snímků* je skupina[2] snímků s několika jasně danými kritérii. Snímky měření jsou v podstatě výsledkem procesu snímkování, jehož příklad byl uveden v kapitole 2.

První ze zmíněných kritérií pro existenci série snímků je existence pořadí snímků, které lze určit dvěma způsoby:

- podle nadefinované hodnoty global Z
- podle abecedy dle názvů snímků

[2] Tato skupina musí mít více než jeden snímek. Nemá cenu uvažovat měření, které nemá žádné snímky anebo pouze jediný snímek

První způsob předpokládá, že každý snímek má danou pozici v prostoru. Tento parametr si můžeme představit jako hodnotu pomyslné osy z, na které snímky řadíme. Popsaný způsob je nejběžnější v reálné praxi a měl by být při určování pořadí snímku preferován.

Druhý způsob používáme v případě, že snímky nemají nastavenu hodnotu global Z. Tento fakt se zpravidla pozná podle toho, že mají všechny snímky defaultní nastavení[3]. Snímky tedy seřadíme abecedně a přisoudíme jim fiktivní hodnotu global Z. Prvnímu snímku zůstane hodnota 0, druhý bude mít hodnotu 1 atd.

Dalším kritériem pro sérii snímků je, že musí obsahovat stejné zájmové objekty. Toto pravidlo pouze zrcadlí fakt, že se snímky měření v praxi vždy soustředí na jednu měřenou oblast. Jestli jsou dva objekty stejné anebo ne, se určuje podle jejich typu a jména.

Shoda musí panovat mezi snímky v sérii také v nastavených měřících a distančních jednotkách.

Při 2D modelování se dále setkáme také s tím, že budeme chtít modelovat parametry dvou sérií navzájem. Tyto série musí splňovat výše uvedená kritéria tak, aby z nich bylo teoreticky možné vytvořit sérii jedinou – tzn., musí navzájem obsahovat stejné zájmové objekty[4], mít nastaveny stejné měřící jednotky a jednotky distance.

4.1.3 Distanc

V předcházející kapitole jsme zmínili pomyslnou prostorovou osu z, na které řadíme snímky. Hodnota, která náleží danému snímku (tedy jeho pozice v prostoru, nebo též vzdálenost v prostoru) se nazývá distanc.

Každý snímek má ve vlastní definici určeno, v jakých jednotkách bude distanc měřen. Může jít buď o jednotky délkové (mm, cm, m) anebo jednotky časové (datum v různých formách). Nicméně jak bylo v předcházející kapitole naznačeno, hodnota distance se určuje dle definované hodnoty global Z snímku a systém Fotom NG ve své stávající verzi podporuje pouze celočíselné hodnoty této proměnné. V další specifikaci se proto omezíme pouze na délkové varianty distance.

4.1.4 Lícovací body a měřící jednotky

Souřadnice zájmového objektu jsou na digitálně zpracovaném snímku uloženy v pixelech. Tyto souřadnice označujeme jako *lokální*. Abychom byli schopni zjistit, jakým skutečným (*globálním*) hodnotám odpovídají, musíme znát příslušné měřítko. To je dáno dvojicí *lícovacích bodů*. U těchto bodů známe jak jejich lokální souřadnice, tak souřadnice globální a navíc také jejich lokální i globální vzdálenost. Díky této informaci jsme schopni vzájemně převádět body v pixelech a body v prostoru.

Snímek si krom lícovacích bodů dále musí nést informaci, v jakých skutečných jednotkách je měřen. Obecně to mohou být milimetry, centimetry nebo metry.

4.1.5 Zájmové objekty a zájmové body

Zjednodušeně řečeno jsou zájmové objekty dvourozměrné obrazce, kterými znázorňujeme skutečné objekty na snímku – např. řez cévou může být označen kružnicí, čtvercový profil domovní fasády polygonem atd. Motivace pro takové zjednodušení byla uvedena v podkapitole 2.2.

Systém Fotom NG umožňuje na snímku definovat celkem 6 typů zájmových objektů. My se ovšem v dalším výkladu omezíme pouze na ty, které jsou *vhodné k 2D modelování*[5].

[3] V případě Fotomu NG je to celočíselná hodnota 0

[4] Z toho samozřejmě plyne, že těchto objektů musí být stejný počet

[5] Vynecháme objekty mřížka a matice, které se nevztahují ke skutečným objektům

Nejjednodušším zájmovým objektem je *bod*. Je dán svou polohou (souřadnicemi) a má nulový obvod i obsah. Bod se často používá pro definici nebo určení polohy složitějších objektů a proto někdy užíváme termínu *zájmový bod*.

Jiným zástupcem zájmových objektů je *průsečík*, který je definován jako jediný společný bod dvou úseček. Každá úsečka je dána právě dvěma body, průsečík je tedy definován právě čtyřmi body. Stejně jako bod má svou polohu v prostoru a neurčujeme u něj obsah ani obvod.

V úvodu zmíněná *kružnice* je množina bodů ve stejné vzdálenosti od jejího středu. Ve Fotomu NG existují dva způsoby, jak kružnici zadat. První je pomocí středu a poloměru. Druhý způsob předpokládá vytvoření polygonu, kterému je pak kružnice opsána. Jako zájmový bod kružnice je v každém případě vnímán její střed. Je nasnadě, že u kružnice má smysl počítat její obvod a obsah plochy, který vytyčuje[6].

Poslední zájmovým bodem vhodným pro 2D modelování je *polygon* neboli česky mnohoúhelník. Jedná se nejobecnější zadání zájmového objektu, který je v tomto případě definován jako uzavřená řetězová spojnice *n* bodů. Zájmovým bodem polygonu je jeho těžiště. Polygon má svůj obvod, daný jako součet vzdáleností jednotlivých spojnic bodů, a stejně tak svůj obsah.

4.1.6 Sledované parametry

Známe-li zájmové objekty, které lze na snímcích definovat, musíme rovněž určit, jaké jejich parametry budeme v průběhu 2D modelování měřit a evidovat.

Je nasnadě, že u každého objektu je možné sledovat jeho *polohu*. Ta je dána buď přímo (u bodu a průsečíku) anebo jako pozice zájmového bodu (střed kružnice, těžiště polygonu).

Speciálně u zájmového objektu kružnice je rovněž možné sledovat její velikost, lépe řečeno velikost jejího *poloměru*.

Na základě obvodu a obsahu kružnice a polygonu můžeme také dopočítávat kumulativní objem a kumulativní obsah pláště trojrozměrného objektu, které tyto objekty vytváří v prostoru. Oba uvedené pojmy budou dále rozebrány v kapitole 4.3.1.

4.1.7 Vzdálenosti, odchylky a projektované hodnoty

Další parametr, který budeme chtít měřit je vzdálenost dvou zájmových objektů na snímku. Matematicky tento vztah vyjádříme jako vzdálenost zájmových bodů těchto objektů.

Kromě sledování změn parametrů můžeme také chtít určit, jak a jestli se tyto parametry odchylují od námi dané hodnoty. Tato hodnota může být dvojího typu:

- aritmetický průměr z hodnot daného parametru na všech snímcích
- uživatelem specifikovaná hodnota (označovaná jako projektovaná)

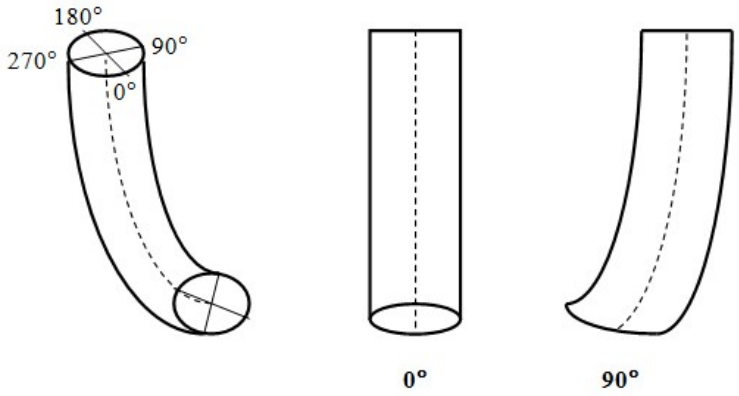
V prvém případě jde o jednoduchý výpočet odchylky od průměru. V případě druhém potřebujeme další snímek, který musí splňovat všechny podmínky tak, aby mohl být teoreticky přidán do právě modelované série – tzn., musí obsahovat stejné zájmové objekty, musí mít nastaveny stejné měřící jednotky a stejné jednotky distance. Hodnotu parametru pro porovnání pak budeme odečítat právě z tohoto snímku a nazývat *projektovanou hodnotou*.

Je zřejmé, že odchylky můžeme počítat nejenom pro sledované parametry, ale také pro vzdálenosti objektů.

[6] Pokud bychom chtěli být matematicky přesní, pak by se samozřejmě jednalo o obsah kruhu, který je kružnicí určen

4.1.8 Úhel pohledu

Mějme důlní jámu s kruhovým profilem a představme si její reprezentaci v prostoru. Pokud je jáma nějakým způsobem zdeformovaná můžeme při pohledu z různých úhlů dojít k různým výsledkům. Nejlépe daný problém ilustruje následující obrázek z práce pana Ing. Koštuříka [8].



Obrázek 5 - Natočení modelu důlní jámy

Jak vidíme, je tvar důlní jámy silně prohnutý, přesto se nám však při pohledu z úhlu 0° bude jevit jako zcela rovný. *Úhel pohledu* tedy určuje natočení objektu v prostoru nebo zjednodušeně úhel, ze kterého se na objekt díváme.

4.1.9 Výpočet

Už dříve jsme mluvili o tom, že budeme chtít sledovat některé parametry, případně dopočítávat jiné, určovat odchylky atd. Funkcionalitu, která provede takový úkon, budeme dále označovat jako *výpočet* neboli kalkulaci.

Výpočty budeme označovat názvy, které budou vystihovat jejich hlavní smysl a pod kterými budou uvedeny v uživatelském rozhraní budoucího modulu.

Výstup kalkulace může být dvojí – grafická reprezentace v podobě jednoho nebo několika grafů anebo jedna nebo více tabulek s vypočtenými hodnotami.

4.1.10 Grafy a tabulky

Grafy, o kterých budeme dále mluvit a které budou využívány pro zobrazení výsledku výpočtů, budou výhradně dvourozměrné. Na horizontální ose při tom budou vždy hodnoty distance. Na ose vertikální budou potom umísťovány parametry, jejichž průběh chceme sledovat.

Každý graf bude opatřen legendou a dodatečnými statickými informacemi jako je průměrná hodnota sledované veličiny a její rozsah (tj. rozdíl maxima a minima).

Tabulky budou vždy obsahovat sloupce pro jméno snímku a distanc. K těmto sloupcům se poté přidají další v závislosti na typu výpočtu. Jeden řádek tabulky znamená jednu hodnotu distance. Budeme-li tedy mít sérii s 5 snímky, bude na horizontální ose grafů umístěno 5 hodnot a tabulky budou mít 5 řádků.

4.2 Vize a obecné požadavky

Navrhovaný modul, který je předmětem této diplomové práce, bude uživateli systému Fotom NG poskytovat veškeré možnosti pro 2D modelování objektů na snímcích. Základem této funkcionality budou výpočty implementované v modulu 2 systému Fotom 2008. Nový modul musí obsahovat veškeré funkce svého předka a dále je rozšiřovat a vylepšovat z hlediska správnosti, výkonu a uživatelské přívětivosti.

Modul bude plně integrován do systému Fotom NG. Bude využívat jeho stávající aplikační i uživatelské rozhraní. Pokud budou pro potřeby navrhovaného modulu vytvářeny nové doplňky uživatelského rozhraní (např. dialogy, nastavení, ovládací panely), pak musí tyto komponenty splňovat funkční i vzhledové zásady stávajícího rozhraní.

Nový modul umožní uživateli otevírat snímky do módu 2D modelování, ve kterém budou považovány za měření (neboli sérii). Tyto snímky budeme nazývat jako *snímky otevřené v sérii*. Série snímku tedy nebude nikde ukládána a nebude vytvářet samostatný objekt. Uživateli bude umožněno přidat do jednoho módu dvě samostatné série, které budou splňovat požadavky popsané v kapitole 4.1.2. Podobným způsobem bude také uživateli umožněno přidat k módu jeden samostatný snímek, jehož parametry budou při výpočtu brány jako projektované hodnoty.

Mód 2D modelování bude uživateli umožňovat spouštět výpočty nad vybranou sérií (sériemi) a nad zvolenými objekty této série (sérií). Výsledky výpočtu budou zobrazeny v grafické podobě a v podobě tabulek. Uživatel bude mít možnost si oba tyto přístupy zobrazit samostatně. Mezi grafem a tabulkou musí existovat interaktivní vazba. Pokud uživatel označí bod v grafu, musí být vyznačen odpovídající záznam v tabulce a naopak.

Uživateli bude umožněno přepínat mezi různými výpočty. V průběhu zobrazení výstupu může uživatel měnit zvolené objekty, sledované parametry nebo úhel pohledu a grafický i tabulkový výstup se musí automaticky náležitě přizpůsobit.

4.3 Výpočty poskytované modulem

Navrhovaný modul bude uživateli poskytovat kolekci výpočtu, které bude moci nad zvolenými daty provádět.

Vstupy pro jednotlivé výpočty se mohou různit. Například některé výpočty potřebují ke svému průběhu dvě série, jiné pouze jednu. Výpočty odchylek zase podporují práci s projektovanými hodnotami, kdežto třeba u výpočtu vzdálenosti objektů zadávání těchto hodnot nedává žádný smysl.

Stejně tak výstupy jednotlivých kalkulací se budou lišit. Přestože víme, že výstupem každého výpočtu má být grafické zpracování a tabulka, počet grafů může být různý a také struktura tabulky nebude vždy stejná.

V následujícím text popíšeme jednotlivé výpočty a pro každý z nich vymezíme povolený vstup (tedy co uživatel může anebo musí nastavit), charakterizujeme princip a smysl výpočtu a nakonec určíme, v jakém tvaru bude jeho výstup (tedy co bude chtít uživatel na konci vidět).

4.3.1 Parametry objektů

Toto je nejjednodušší typ výpočtu a jeho hlavním smyslem je zobrazit průběh změn zvoleného parametru v průběhu distance. Mějme například sérii snímků, kde na každém z nich je definován zájmový objekt kružnice. Naším sledovaným parametrem může být v tomto případě například plocha. Vypočteme přesné hodnoty plochy pro kružnice na jednotlivých snímcích a umístíme je do grafu. Úkolem výpočtu je interpolovat křivku mezi známými hodnotami a zobrazit tak, jak se obsah plochy mění v závislosti na vzdálenosti v prostoru.

Potřebný vstup můžeme popsat následující tabulkou:

Počet zpracovávaných sérií	1
Počet zvolených objektů	1 a více
Sledované parametry	ano
Úhel pohledu	ne
Projektované hodnoty	ne

Tabulka 1 - Charakterizace vstupu výpočtu Parametry objektů

Výpočet tedy pracuje pouze s jednou sérií snímků. Uživatel může sledovat tentýž parametr na více objektech, minimálně však na jednom. V grafu se v tomto případě zobrazí více křivek, každá pro jeden objekt. Už jsme zmínili, že tento typ výpočtu vyžaduje po uživateli, aby zvolil sledovaný parametr. Jaké typy parametrů má k dispozici, je popsáno v podkapitole 4.1.6.

Výstup kalkulace bude závislý na tom, jaký sledovaný parametr uživatel zvolil. Bude-li to střed (těžiště) objektu, pak budou výstupem dva grafy. První bude zobrazovat průběh x-ové souřadnice zájmového bodu a druhý souřadnice y-ové. U všech ostatních sledovaných parametrů půjde vždy o jeden graf.

Ve speciálním případě, kdy uživatel nastaví jako sledovaný parametr střed (těžiště), zvolí pouze jediný zájmový objekt a tento objekt bude typu kružnice, přidá se ke standardním dvěma grafům ještě třetí, který bude zobrazovat průběh změny poloměru této kružnice.

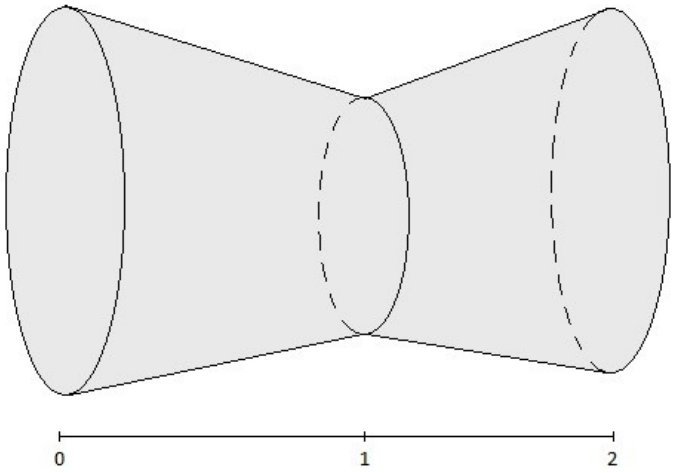
Tabulka, generovaná výpočtem, bude mít krom pevně daných sloupců (viz. podkapitola 4.1.10) také další sloupce pro sledované parametry. V případě, že je sledovaným parametrem střed (těžiště), přibudou dva sloupce pro x-ové a y-ové souřadnice zájmového bodu. Ve výše zmíněném speciálním případě se přidá ještě jeden sloupec, který bude obsahovat hodnoty poloměru. Při zvolení ostatních typů sledovaných parametrů, pak přibude vždy jeden sloupec s vypočtenou hodnotou parametru.

Pokud uživatel označí více objektů, pak bude výstupem výpočtu také více tabulek. Každá pak musí být jednoznačně označena jménem zájmového objektu, jehož výsledky zobrazuje.

Zastavme se ještě u sledovaných parametrů kumulovaný objem a obsah pláště. Způsob jejich výpočtu nemusí být na první pohled zřejmý a tak jej zde blíže vysvětlíme. V úvodu této kapitoly byl zmíněn příklad se zájmovou kružnicí a její plochou. Pokud si představíme, že jednotlivé snímky jsou řezy nějakého objektu, můžeme určit jeho objem v jednotlivých distancích. Kumulovaný objem na prvním snímku bude rovný nule. Na druhém snímku jej spočteme jako objem komolého kužele, kde zájmová kružnice prvního a druhého snímku tvoří jeho podstavy a výška je dána vzdáleností mezi snímky (tedy distancem). Kumulovaný objem na třetím snímku určíme obdobně (podstavy tvoří zájmové kružnice na druhém a třetím snímku) avšak nesmíme zapomenout přičíst objem z předchozího kroku[7].

Popsaný postup můžeme samozřejmě aplikovat také na zájmový objekt polygon. Pokud budeme místo plochy počítat s obvody zájmového objektu, pak budou výsledné hodnoty tvořit kumulovaný obsah pláště.

[7] Jednotlivé objemy se sčítají (proto mluvíme o kumulativním objemu) a při posledním snímku tak budeme znát celkovou hodnotu objemu tělesa, jehož řezy snímky tvoří



Obrázek 6 - Prostorový objekt, jehož řezy se na snímcích zobrazí jako kružnice

4.3.2 Vzdálenosti mezi objekty

Úkolem tohoto výpočtu je zobrazit, jak se mění vzdálenost mezi dvěma objekty na snímcích. V rámci výpočtu se vytvoří pomyslná spojnice mezi zájmovými body objektů (středy, těžišti nebo samotnými body) a její délka udává hledanou vzdálenost. Krom toho vyjadřuje výpočet také odklon této pomyslné spojnice od osy x.

Vstup výpočtu můžeme popsat následující tabulkou:

Počet zpracovávaných sérií	1
Počet zvolených objektů	2
Sledované parametry	ne
Úhel pohledu	ne
Projektované hodnoty	ne

Tabulka 2 - Charakterizace vstupu výpočtu Vzdálenosti mezi objekty

Výpočet vyžaduje pouze jedinou sérii a právě dva zvolené objekty. Veličiny, které nás zajímají, jsou pevně dány (vzdálenost a odklon) a výpočet tedy po uživateli nepožaduje žádné jiné zadání sledovaného parametru.

Z výše uvedených faktů můžeme odvodit, že výpočet vykreslí vždy dva grafy – první bude zobrazovat průběh vzdálenosti a druhý měnící se odklon. Každý z těchto grafů bude mít pouze jedinou křivku.

Podobně u tabulky přibudou právě dva sloupce s hodnotami – jeden pro vzdálenost a jeden pro odklon od osy x.

4.3.3 Odchylky parametrů objektů

Zobrazení odchylek znamená vykreslení průběhu změn parametrů vzhledem k nějaké dané hodnotě. O tom, jaká tato hodnota může být, jsme určili v podkapitole 4.1.7.

Podívejme se nejprve, jak lze popsat vstup výpočtu:

Počet zpracovávaných sérií	1
Počet zvolených objektů	1
Sledované parametry	ano
Úhel pohledu	ne
Projektované hodnoty	ano

Tabulka 3 - Charakterizace vstupu výpočtu Odchylky parametrů objektů

Opět budeme pracovat s jednou sérií snímků. Uživatel může zadat pouze jediný objekt, nad kterým má být počítáno. Výstupy výpočtu se totiž mohou pro jednotlivé objekty výrazně lišit a nebylo by je možné umístit do jediného grafu tak, jak tomu bylo při parametrech objektů. Protože sledujeme odchylky nějaké veličiny, bude po uživateli požadováno zadání konkrétního sledovaného parametru. Kalkulace odchylek rovněž povoluje zadání projektovaných hodnot. Logika výpočtu je jednoduchá – pokud máme tyto hodnoty k dispozici, pak počítáme odchylky vůči nim. Pokud projektované hodnoty k dispozici nejsou, pak nejprve vypočítáme aritmetický průměr hodnot parametru a pak určujeme odchylky vůči němu.

Výstupem tohoto výpočtu bude kolekce grafů, jejichž počet bude závislý na typu sledovaného parametru, podobně jako tomu bylo v případě parametrů objektů (viz. 4.3.1). Na vertikální ose grafů budou umísťovány hodnoty odchylky. Je zřejmé, že v hodnotě 0 je odchylka nulová a tedy parametr je přesně stejný jako aritmetický průměr nebo projektovaná hodnota. Nicméně se nedovíme, jak velká je tato hodnota. Proto je zapotřebí horizontální osu speciálně vyznačit a popsat hodnotou průměru (projektované hodnoty), vůči které odchylky počítáme. Tento úkon nám zároveň rozdělí graf na dvě poloviny a bude tak od prvního pohledu zřejmé, v jakém distanci je odchylka menší a v jakém naopak větší než průměrná (projektovaná) hodnota.

Pro větší názornost bude také vhodné přidat průměr či projektovanou hodnotu rovnou do popisku grafu. V případě projektovaných hodnot musíme také přidat název souboru, ze kterého jsou tyto hodnoty odečítány.

Zmiňme se na tomto místě, jak se počítá projektovaná hodnota kumulovaného objemu a obsahu pláště. Může se totiž zdát, že to není možné, protože máme jenom jediný snímek a podle postupu popsanému v podkapitole 4.3.1 z něj kumulovaný objem nelze vypočíst. Řešení je následující: vytvoříme fiktivní sérii snímků, která bude mít stejné rozestupy distance jako série, nad kterou provádíme výpočet. V těchto distancích se však bude nalézat zas a znovu právě ten jediný snímek, ze kterého odečítáme hodnoty. Průměrná hodnota kumulovaného objemu této fiktivní série nám potom poslouží jako projektovaná hodnota. Analogicky můžeme domyslet výpočet projektované hodnoty pro obsah pláště.

Tabulka, která bude výstupem této kalkulace, bude obsahovat oproti standardním sloupcům další sloupce pro odchylky. V případě, že sledovaným parametrem bude střed (těžiště), pak to budou dva sloupce pro odchylku souřadnice X a pro odchylku souřadnice Y. I zde zohledňujeme to, zda je zvoleným zájmovým objektem kružnice. Pokud ano, přidáme další sloupec pro odchylku poloměru. U všech zbývajících sledovaných parametrů přibude vždy jeden sloupec.

4.3.4 Odchylky vzdáleností objektů

Princip výpočtu odchylek vzdáleností je velmi podobný předchozímu výpočtu odchylek parametrů avšak s tím rozdílem, že zde se soustředíme na veličiny vzdálenosti a odklonu spojnice objektů od osy x. Abychom mohli vypočítat vzdálenost, potřebujeme na vstupu kalkulace dva objekty, jak lze vyčíst také z popisu vstupu:

Počet zpracovávaných sérií	1
Počet zvolených objektů	2
Sledované parametry	ne
Úhel pohledu	ne
Projektované hodnoty	ano

Tabulka 4 - Charakterizace vstupu výpočtu Odchylky vzdáleností

Výstupem výpočtu budou dva grafy – jeden pro odchylky vzdáleností a jeden pro zobrazení odchylek odklonu spojnice. I pro tyto grafy potřebujeme vyznačit horizontální osu a uvést v popisku grafu konkrétní hodnotu, případně název souboru, ze kterého odečítáme projektované hodnoty.

Jako projektovanou hodnotu ze snímku považujeme jednoduše vzdálenost zvolených dvou objektů na tomto snímku. Stejně určíme projektovaný odklon spojnice od osy x. Průměrné hodnoty určujeme ze vzdáleností (odklonů) na jednotlivých snímcích.

Výstupní tabulka bude obsahovat navíc dva sloupce pro odchylku vzdáleností a odklonu spojnic.

4.3.5 Porovnání dvou měření – parametry

Jak napovídá název, slouží tento typ výpočtu ke vzájemnému porovnání průběhu změn zvoleného parametru ve dvou sériích snímků. Tento postup si můžeme představit tak, že provedeme výpočet označený jako Parametry objektů pro obě dvě série a pak jednotlivé grafy sjednotíme. Křivky přitom označíme sufixem, který bude udávat, zda se jedná o první nebo o druhé měření.

Charakterizujme potřebný vstup výpočtu:

Počet zpracovávaných sérií	2
Počet zvolených objektů	1
Sledované parametry	ano
Úhel pohledu	ne
Projektované hodnoty	ne

Tabulka 5 - Charakterizace vstupu výpočtu Porovnání dvou měření – parametry

Jak je vidět, volíme vždy pouze jediný zájmový objekt. Ve výsledném grafu pro daný parametr tak budeme mít právě dvě křivky, kde jedna bude označovat první měření a další druhé. Je samozřejmé, že uživatel musí zvolit, který konkrétní parametr chce sledovat.

Výstupem kalkulace bude kolekce grafů, jejichž počet opět závisí na zvoleném sledovaném parametru, případně také na tom, zda je vybraný zájmový objekt typu kružnice.

Také tabulka bude sjednocením dvou tabulek, které dostaneme z výstupu výpočtu parametrů objektů. I zde musíme jednotlivé sloupce opatřit sufixem s označením prvního nebo druhého měření.

4.3.6 Porovnání dvou měření – vzdálenosti

V tomto výpočtu (obdobně jako v předchozím případě) opět sjednotíme výstupy dvou dílčích výpočtů. Tentokrát se ale bude jednat o výpočty vzdáleností objektů.

Určeme, jak by měl vypadat vstup výpočtu:

Počet zpracovávaných sérií	2
Počet zvolených objektů	2
Sledované parametry	ne
Úhel pohledu	ne
Projektované hodnoty	ne

Tabulka 6 - Charakterizace vstupu výpočtu Porovnání dvou měření – vzdálenosti

Není žádným překvapení, že potřebujme znát přesně dva objekty. Výstupem kalkulace budou samozřejmě dva grafy – jeden bude zobrazovat průběh vzdáleností pro první a druhé měření a druhý vykreslí totéž pro odklony spojnic od osy x. Také tabulka vznikne analogií k postupu popsanému v předchozí kapitole.

4.3.7 2D model objektu

Tento speciální typ výpočtu je určen k zohlednění relativního natočení objektu – jinými slovy bere v potaz, z jakého úhlu pohledu se na objekt díváme.

Vstup výpočtu popisuje následující tabulka:

Počet zpracovávaných sérií	1 nebo 2
Počet zvolených objektů	1
Sledované parametry	ne
Úhel pohledu	ano
Projektované hodnoty	ne

Tabulka 7 - Charakterizace vstupu výpočtu 2D model objektu

Zvláštností oproti dříve uvedeným kalkulacím je, že výpočet můžeme uplatnit jak na jednu, tak na dvě série. Protože volíme právě jeden objekt, bude v grafu pokaždé jedna nebo dvě křivky – pro jedinou nebo pro obě série. Připomeňme, že v tomto případě také požadujeme, aby uživatel zadal úhel pohledu ve stupních. Pokud tak neučiní, volíme úhel 0°.

Výstupem kalkulace budou čtyři grafy. Prvé budou zobrazovat průběh parametru střed (těžiště), tedy konkrétně průběh souřadnic X a Y. Rozdíl oproti běžnému výpočtu parametrů je v tom, že kalkulace nejprve každý bod otočí o daný úhel a teprve pak zobrazí jeho hodnoty do grafu.

Druhé dva grafy budou zobrazovat 2D reprezentace prostorový modelů. Tento model vznikne tak, že každý bod souřadnicové křivky posuneme o polovinu plochy objektu v daném distanci. Nejprve polovinu přičteme a poté odečteme. Vzniknou nám tak dvě křivky, kde jedna bude tvořit horní mez modelu a druhá dolní. Prostor mezi těmito křivkami vyplníme barvou křivek a vznikne nám tak plocha, která bude reprezentovat zakřivení trojrozměrného objektu v prostoru.

Pokud budeme mít na vstupu výpočtu k dispozici dvě série, budou samozřejmě v grafu modelu čtyři křivky, které budou vytvářet dvě plochy. Je třeba dbát na to, aby bylo vidět, jak se navzájem překrývají a aby se nestalo, že by jedna úplně zakryla druhou.

Výstupní tabulka výpočtu bude tvořena standardními sloupci a pak dvojici dalších pro hodnoty průběhu x-ové a y-ové souřadnice.

4.4 Ostatní funkcionalita

Krom samotných výpočtů a vyváření sérií bude modul poskytovat dodatečné funkce pro práci s grafy a tabulkami. Požadavky na tyto funkce jsou specifikovány v následujících podkapitolách.

4.4.1 Nastavení zobrazení grafu

Jednotlivé křivky v grafech budou zobrazovány různými barvami. Která barva přísluší konkrétnímu zájmovému objektu, bude uvedeno pod každým grafem v legendě. Uživateli musí být dána možnost nastavit si číselník barev, podle kterého budou barvy pro křivky nastavovány.

V grafu budou pro každý bod označeny horizontální i vertikální vynášecí (vodící) čáry a u každého bodu bude uvedena hodnota, kterou reprezentuje. Tato hodnota musí být v barvě příslušné křivky. Uživatel může vodící čáry i zobrazení hodnot kdykoliv zakázat nebo opět povolit.

Grafy by měly obsahovat funkci pro přiblížení (zoom).

4.4.2 Aktivace / deaktivace sérii

Pokud uživatel přidá do módu 2D modelování dvě série, případně soubor, že kterého budou odečítány projektované hodnoty, musí mít možnost kdykoliv deaktivovat druhou sérii nebo zmíněný soubor. Výpočet se pak bude chovat tak, jako by v něm vůbec ona série nebo soubor nebyly zadány.

Tato funkcionalita slouží k tomu, aby se uživatel mohl vrátit do stavu, kdy má otevřenu jednu sérii i potom, co k modelování přidal druhou. Série není z modelování úplně odstraněna, pouze je její přítomnost v daný moment přehlížena.

Aktivací a deaktivací souboru s projektovými hodnotami může uživatel snadno přepínat mezi odchylkami od průměru a odchylkami od projektovaných hodnot.

4.4.2 Filtrování tabulky hodnot

Poblíž tabulky hodnot bude umístěn formulář, který umožní její filtraci. Uživatel bude moci zadat horní a dolní mez pro filtrovaná data. Hodnoty v tabulce budou filtrovány vždy pouze podle jednoho sloupce. Modul musí přehledně vyznačit, zda je filtr právě aktivní nebo ne.

Data budou moci být vždy filtrována podle distance. Další možnosti filtrace budou závislé na tom, jaké hodnoty jsou právě zobrazovány (tj. jaký výpočet je právě prováděn). Jinými slovy – pokud je právě zobrazena tabulka s hodnotami sledovaného parametru střed (těžiště), pak bude filtr uzpůsoben pro zadání horní a dolní meze souřadnic X anebo Y. Pokud je sledovaný parametrem plocha, pak lze do formuláře filtru zadat pouze meze pro obsah.

4.4.3 Export

Uživatel bude moci exportovat tabulky s výstupními daty. Bude mít při tom na výběr z následujících formátů:

- CSV
- XLS
- XLSX

CSV značí odřádkovaný seznam v textovém souboru, kde jsou jednotlivé položky odděleny čárkou. Přesná pravidla pro vzhled a formátování výstupních dat jsou dána standardem [16].

Soubory typu XLS a XLSX jsou dobře známy díky populárním systémům Microsoft Excel a OpenOffice. Starší formát XLS byl využíván ve verzích programu Excel 97 až 2003 a poté byl nahrazen formátem XLSX, který je založen na standardech OOXML. Navrhovaný modul by měl zvládnout oba druhy exportů, aby bylo dosaženo zpětné kompatibility.

4.4.4 Tisk

Nový modul bude umožňovat vytisknout právě zobrazenou kolekci grafů. Bude obsahovat vhodně umístěné a označené tlačítko, po jehož aktivaci se otevře standardní dialog pro tisk dle používaného operačního systému a tiskárny.

Grafy se budou tisknout v podobě, jaká je právě nastavena. To znamená, že pokud uživatel například zakáže zobrazování vodících čar, pak nebudou tyto čáry vidět ani ve vytištěné podobě.

4.5 Softwarové prostředky

Aby mohl být navrhovaný modul integrován do stávajícího systému Fotom NG, je nutné, aby byl implementován jako zásuvný modul v prostředí NetBeans Platform. O hlavních rysech a výhodách této platformy pojednává kapitola 4.5.1.

Z výše specifikovaných požadavků je také zřejmé, že jedním ze základních kamenů modulu bude proces zobrazování a zpracovávání grafů. Zatímco výstupní tabulky lze snadno realizovat pomocí standardní javovské komponenty *JTable*, pro práci s grafy Java žádné podobné prostředky nenabízí. Bylo tedy potřeba vyhledat nějakou volně přístupnou knihovnu, která by splňovala následující požadavky:

- používání knihovny musí být bezplatné
- knihovna musí podléhat licenci, která povoluje její bezvýhradnou integraci do jiných (nekomerčních) projektů
- grafy musí obsahovat takové možnosti a funkcionalitu, aby byly schopné vyhovět požadavkům kladeným na grafickou podobu výstupu výpočtů
- vzhled výsledných grafů musí podléhat určitému kvalitativnímu standardu a modernímu způsobu zpracování (například musí podporovat interpolaci polynomem vyššího řádupo částech, tedy interpolaci křivkou spline)

Těmto kritériím nejlépe vyhověla knihovna JFreeChart. Její popis je obsahem podkapitoly 4.5.2.

Dalším požadavkem bylo umožnění exportu výstupních dat do různých formátů. Zatímco formáty XLSX a CVS jsou textové a jejich specifikace veřejně přístupné, vytvoření souboru XLS pomocí standardních prostředků jazyka Java by mohlo znamenat problém. Ostatně i v případě předchozích dvou formátů by bylo vhodnější spolehnout se na nějakou externí knihovnu s prověřenou funkcionalitou. Jedním ze zástupců takové knihovny je Apache POI, kterému se věnuje podkapitola 4.5.3.

4.5.1 NetBeans Platform

NetBeans Platform je modulární a rozšiřitelný aplikační framework pro vývoj RCP. Poskytuje vývojářům architekturu, základní funkcionalitu a připravená řešení pro vývoj rozsáhlých Swing aplikací. Vývojáři se tak mohou soustředit na hlavní myšlenky svých systémů a nemusí řešit dílčí úkoly jako je připojování akcí k menu, vytváření toolbarů, správu oken atd.[17] Nejznámější aplikací napsanou nad platformou NetBeans je samotné NetBeans IDE.

Jednou z hlavních charakteristik je přísná modularita systému. Každá jednotlivá část systému je považována za modul, který má definovány závislosti na jiných modulech a který s nimi komunikuje díky vystavenému API. Moduly jsou dynamicky načítány jádrem platformy, které je rovněž známo jako běhový kontejner [9]. Informace o modulu a jeho závislosti jsou pro každý modul deklarativně zadány v jeho manifestu. Jednoduchý příklad manifestu je zobrazen na následujících řádcích:

```
Manifest-Version: 1.0
OpenIDE-Module: org.app.mymodule
OpenIDE-Module-Layer: org/app/mymodule/layer.xml
OpenIDE-Module-Specification-Version: 1.0
```

Kromě něj obsahuje každý modul také konfigurační soubor layer.xml. V tomto souboru je popsáno vše, co modul přidává do aplikace. Zatímco o manifestu můžeme zjednodušeně říct, že upozorňuje na existenci modulu, layer.xml říká, co modul vlastně umí. Deklarace v layer.xml může například přidat položku do hlavního menu aplikace a určit, která obslužná třída se má spustit po kliknutí. Tento kód může vypadat například takto:

```
<filesystem>
 <folder name="Actions">
  <folder name="SomeMenu">
   <file name="org-app-SomePackage-myclass-MyClassAction.instance">
    <attr name="delegate"
      newvalue="org.app.somepackage.myclasss.myClassAction"/>
    <attr name="noIconInMenu" boolvalue="false"/>
   </file>
  </folder>
 </folder>
</filesystem>
```

Nejdůležitější vlastností souborů layer.xml je, že vytváří strom vzájemně vnořených deklarací. Po spuštění aplikace načte platforma NetBeans všechny soubory layer.xml ze všech modulů a spojí je dohromady. Jednotlivé deklarace se sjednotí a definuje se tak vzhled a chování výsledného systému.

Z výše uvedeného rovněž vyplývá, že modul je samostatný logický a uzavřený prvek, který může být do aplikace snadno přidán nebo odebrán aniž by bylo nutné jakýmkoliv způsobem zasáhnout od kódu nebo ovlivnit funkčnost ostatních modulů.

Další nosnou funkcionalitou platformy NetBeans jsou API, které řeší například:

- komunikaci mezi moduly
- vytváření a správu akcí
- práci s okny aplikace
- práci se soubory ve virtuálním souborovém systému platformy

Uvedená API dále stručně popíšeme.

Lookup API

Lookup API poskytuje prostředky pro komunikaci mezi moduly. Protože mezi moduly neexistují žádné proxy třídy nebo jiné programovatelné přístupy, bylo potřeba vyvinout prostředek, kterým by si moduly mohly předávat informace.

Tímto prostředkem je poskytování lookupu. Lookup je mechanismus, který si můžeme představit jako strukturu Map, kde je klíčem objekt *Class* (např. MojeTrida.class) a hodnotou jsou instance této třídy [1]. Moduly, které chtějí sdílet nějakou informaci, se zaregistrují jako poskytovatelé služby. I toto se provádí deklarativní cestou a to v adresáři META-INF/services v adresářové struktuře modulu. Vytvoří se soubor, který se bude jmenovat stejně jako zmiňovaná služba a do tohoto souboru umístíme informaci o tom, která třída se má zavolat, pokud se po výstupu služby někdo ptá.

Toto doptávání lze provést pomocí třídy *Lookup* z Lookup API:

```
MojeSluzba m = Lookup.getDefault().lookup(MojeSluzba.class);
```

Výše popsaný způsob se uplatní při použití globálního lookupu. Rozeznáváme ale také lookup kontextový, který je závislý na aktuálním kontextu aplikace (ten je určen zpravidla aktuálním oknem aplikace) nebo lookup objektů. Jednotlivé třídy mohou totiž implementovat příslušná rozhraní a poskytovat tak svůj vlastní samostatný lookup.

Mechanismus lookupu podporuje posluchače, takže je možné si na příslušné lookupy zaregistrovat třídy, které budou reagovat na změny v lookupu. Je ale důležité upozornit, že se jedná skutečně o změny obsahu lookupu a ne o vnitřní změny instancí, které jsou právě v daném lookupu vyhledatelné.

Action API

V úvodním příkladu jsme přidali do menu novou položku a deklarovali třídu, která se zavolá po kliknutí. Tato třída byla třídou akce a implementovala rozhraní z Action API. Toto API výrazně rozšiřuje standardní architekturu akcí jazyka Java a umožňuje mimo jiné například vytvářet akce, jejichž chování je závislé na kontextu aplikace.

Akce se registrují díky layer.xml a je výhodou, že lze jedinou akci použít na více místech aplikace. Tedy například není nutné psát dvě rozdílené implementace pro položku hlavního menu a položku lokální nabídky, které mají plnit tutéž funkci. V layer.xml můžeme akcím rovněž nadefinovat klávesové zkratky.

Windows System API

Jak název napovídá, je toto rozhraní zodpovědné za práci s okny aplikace. Všechna okna jsou potomky třídy *TopComponent* a každému z nich je dán tzv. mód [9]. Uspořádání oken, deklarace módu a příslušnost oken k módům jsou dány deklarativně v souboru layer.xml. Mód určuje, kam má být okno primárně dokováno. Rozeznáváme několik typů základních módů:

- editor (hlavní část obrazovky)
- explorer (levý panel)
- output (spodní panel)

Vývojáři je samozřejmě umožněno vytvářet a používat vlastní módy.

Třída *TopComponent* poskytuje každému svému potomku možnost nacházet se během své existence v různých stavech:

- otevřený
- uzavřený
- viditelný (je navrchu)
- neviditelný
- aktivní (má fokus)
- neaktivní

Při změně stavu je automaticky vyvolána událost a okna tak můžou na změny reagovat. Dalšími možnostmi je snadné vytváření kontextových nabídek a perzistence, tedy ukládání stavu okna pro další spuštění aplikace. Tato vlastnost umožňuje uživateli nastavit si vlastní rozmístění oken a platforma NetBeans zajistí, aby bylo zachováno.

Všechny okna jsou dostupná skrze seznam v *TopComponent.Registry* a jejich stav a aktuální mód spravuje dozorčí třída *WindowManager*.

Nodes API
Platforma NetBeans poskytuje API pro jednotlivé úrovně práce se soubory [9]:

- fyzická vrstva
- abstrakční vrstva (File Systém API)
- logická vrstva (Data Systém API)
- prezentační vrstva (Nodes API)

V rámci abstrakční vrstvy jsou data načítána z různých zdrojů (fyzická data, XML reprezentace), v logické vrstvě jsou rozdělena podle typu dat a práce s nimi a konečně prezenční vrstva má za úkol reprezentovat soubory v uživatelském rozhraní.

Uzly (nody) mohou být prezentovány v průzkumníku nebo nastavovány pomocí dialogu vlastností. Můžeme na ně registrovat akce anebo je umisťovat do struktur, jako jsou kontejnery uzlů a vytvářet tak virtuální adresářové hierarchie souborů.

4.5.2 Knihovna JFreeChart

Knihovna JFreeChart je volně dostupná knihovna grafů, která umožňuje vývojářům snadné zobrazení profesionálně vyhlížejících grafů v aplikacích [11]. Jejími hlavními výhodami jsou:

- ucelená a velmi dobře vypracovaná dokumentace [12]
- podpora mnoha typů grafů a zobrazovacích možností
- snadno rozšiřitelná implementace
- rozšířenost a dobrá podpora ze strany vývojářů i uživatelů
- množství ukázkových příkladů
- možnost využití v desktopovém i webovém prostředí
- podpora Swing komponent
- integrované možnosti exportu obrázků do rastrových i vektorových formátů (např. PNG, JPEG nebo PDF, EPS atd.)
- knihovna je distribuována pod licencí LGPL

Princip práce s knihovnou je velmi jednoduchý a dá se shrnout do několika kroků [13]:

1. Vytvoření zdroje dat
2. Vytvoření objektu JFreeChart, který bude zodpovědný za vykreslení grafu
3. Vykreslení grafu do nějakého výstupu (nejčastěji *ChartPanel*)

Zdroj dat je v rámci JFreeChart nazýván *dataset*. Existuje velké množství tříd odvozených od abstraktní třídy *AbstractDataset*, které reprezentují různé druhy dat. V následujícím příkladu je znázorněno, jak lze snadno vytvořit dataset pro koláčový graf:

```
DefaultPieDataset dataset = new DefaultPieDataset();
dataset.setValue("Category 1", 43.2);
dataset.setValue("Category 2", 27.9);
dataset.setValue("Category 3", 79.5);
```

Existuje ještě jeden způsob, jak vytvořit dataset a to pomocí kolekce křivek. Nejprve musíme vytvořit křivky a naplnit je daty:

```
XYSeries series1 = new XYSeries("Series 1");
series1.add(2,56);
series1.add(3,41);
series1.add(4,32);

XYSeries series2 = new XYSeries("Series 2");
series2.add(2,13);
series2.add(3,15);
series2.add(4,17);
```

A poté z nich vytvořit kolekci:

```
XYSeriesCollection seriesCollection = new XYSeriesCollection();
seriesCollection.add(series1);
seriesCollection.add(series2);
```

U vybraných grafů[8] lze kolekci křivek použít místo datasetu. Tento způsob je také používán v implementaci navrhovaného modulu.

Máme-li připravená data, můžeme přistoupit k vytvoření objektu grafu. I to lze provést několika způsoby, nejsnazší je však použít statickou třídu *ChartFactory*, které je navržena dle návrhového vzoru Factory a vytvoří nám graf podle předaných požadavků:

```
JFreeChart graph = ChartFactory.createXYLineChart(("XSplineRenderer",
         "X", "Y", null, PlotOrientation.VERTICAL, true, false, false);
```

Metodě třídy *ChartFactory* předáváme:

- název grafu
- popis x-ové osy
- popis y-ové osy
- dataset (v tomto případě je to *null*, ukážeme si předání dat pomocí kolekce křivek)
- orientaci (v tomto případě je to klasický vzhled – osa x je horizontální)
- zda chceme zobrazit legendu
- zda chceme používat tooltipy
- zda chceme používat URL v exportech používajících HTML image maps

Výsledkem je objekt, který zastupuje graf. Tomuto objektu můžeme přidávat různá nastavení, datasety a jednu z nejdůležitějších věcí – *renderery*. Renderer je třída odvozená od abstraktního předka *AbstractRenderer*, která určuje, jak přesně mají být data z datasetu vykreslena ve finálním grafu. V následující ukázce nejprve přidáme do grafu dříve vytvořenou kolekci křivek a poté této kolekci nastavíme renderer, který zaručí, že křivky budou vykresleny křivkou spline[9]:

```
XYPlot plot = graph.getXYPlot();
plot.setDataset(0, seriesCollection);

XYSplineRenderer renderer = new XYSplineRenderer();
renderer.setSeriesPaint(0, Color.RED);
renderer.setSeriesPaint(1, Color.BLUE);
```

[8] Je zřejmé, že jsou to právě ty grafy, které zobrazují křivky

[9] Máme na mysli, že budou jednotlivé body proloženy křivkou a ne jen spojeny úsečkami. Propojení úsečkami obstarává XYLineRenderer, který je každému křivkovému grafu nastaven defaultně

```
plot.setRenderer(0, renderer);
```

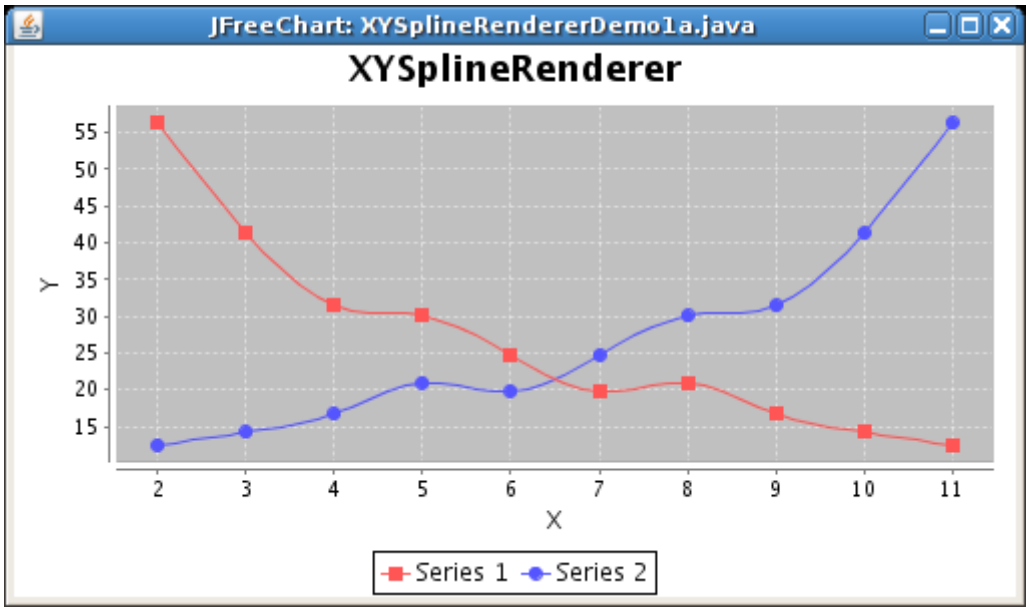
Nejprve jsme museli z grafu vyčíst instanci třídy *XYPlot*, která zaštiťuje všechny datasety a renderery. Do této instance jsme na první pozici přidali naši kolekci křivek. Novému rendereru jsme nastavili, aby vykresloval první křivku v datasetu červeně a druhou modře. Poté jsme přidali renderer do instance třídy *XYPlot* na stejnou pozici jako dříve dataset. To nám zaručí, že právě tento dataset bude vykreslován naším novým rendererem.

Renderery nabízí širokou škálu možností pro vykreslování grafu jako je například zobrazování popisků dat. Díky tomu, že můžeme párovat datasety a renderery, můžeme mít v grafu více druhů křivek.

Poslední částí vytváření grafu je jeho vykreslení. Jako cílová komponenta (tedy kam se má graf vykreslit) se často používá instance třídy *ChartPanel*, který je odvozený od standardní javovské třídy *JPanel*. Následující kód ukazuje, jakým způsobem lze vykreslit data z objektu grafu s daným nastavením do instance *ChartPanel*:

```
ChartPanel panel = new ChartPanel(graph);
```

Pokud bychom použili větší množství dat, pak výsledný graf mohl vypadat například takhle:



Obrázek 7 - Ukázka grafu vytvořeného pomocí knihovny JFreeChart [13]

4.5.3 Apache POI

Projekt Apache POI si vzal za úkol vytvořit a vyvíjet API pro jazyk Java, které by umožňovalo manipulaci se soubory formátů na bázi OOXML a OLE2 [15].

Díky přehledné kolekci tříd a rozhraní je možné velmi snadno vyvářet, číst nebo upravovat soubory na programátorské bázi. Uveďme krátký příklad, který ilustruje vytvoření XLSX souboru a přidání jedné buňky s daty:

```
XSSFWorkbook workbook = new XSSFWorkbook();
XSSFSheet worksheet = workbook.createSheet();
XSSFRow row = worksheet.createRow(0);
XSSFCell column = row.createCell(0);
column.setCellValue(1.23);
```

Hlavní nevýhodou knihovny je její velikost, která značně navyšuje celkovou velikost aplikace, ke které je připojena. Celá knihovna se skládá z několika oddělených archívů, avšak díky jejich vzájemné provázanosti není možné žádný vynechat. Přesto z tohoto plyne určité pozitivum – připojením knihovny získá aplikace možnost kdykoliv v budoucnu využívat její kompletní funkcionalitu a je tedy připravena pro případná další rozšíření v oblasti práce se soubory.

5. Analýza požadavků a návrh modulu

V předchozí kapitole jsme specifikovali požadavky, které jsou na nový modul pro 2D modelování kladeny. Tyto požadavky je nyní nutné analyzovat a vytvořit model navrhovaného systému. Modelem rozumíme přesný, úplný a bezesporný obraz studované reality.

Na základě specifikace požadavků vytvoříme celistvý náhled na systém. Popíšeme jeho chování a přesně specifikujeme rozvrstvení funkcionality. Poté analyzujeme API stávajícího systému a určíme, které prvky je možno nebo nutno využít při implementaci modulu pro 2D modelování. Protože je jedním z požadavků také začlenění GUI modulu do GUI stávajícího systému, navrhneme vhodnou integraci.

Abychom plně využili modularity NetBeans Platform, rozvrhneme funkcionalitu modulu pro 2D modelování do několika dílčích částí, které budou samy v rámci platformy reprezentovány jako moduly. Nejprve navrhneme jejich vzájemné závislosti a způsob komunikace a pak také jejich strukturu.

5.1 Model systému

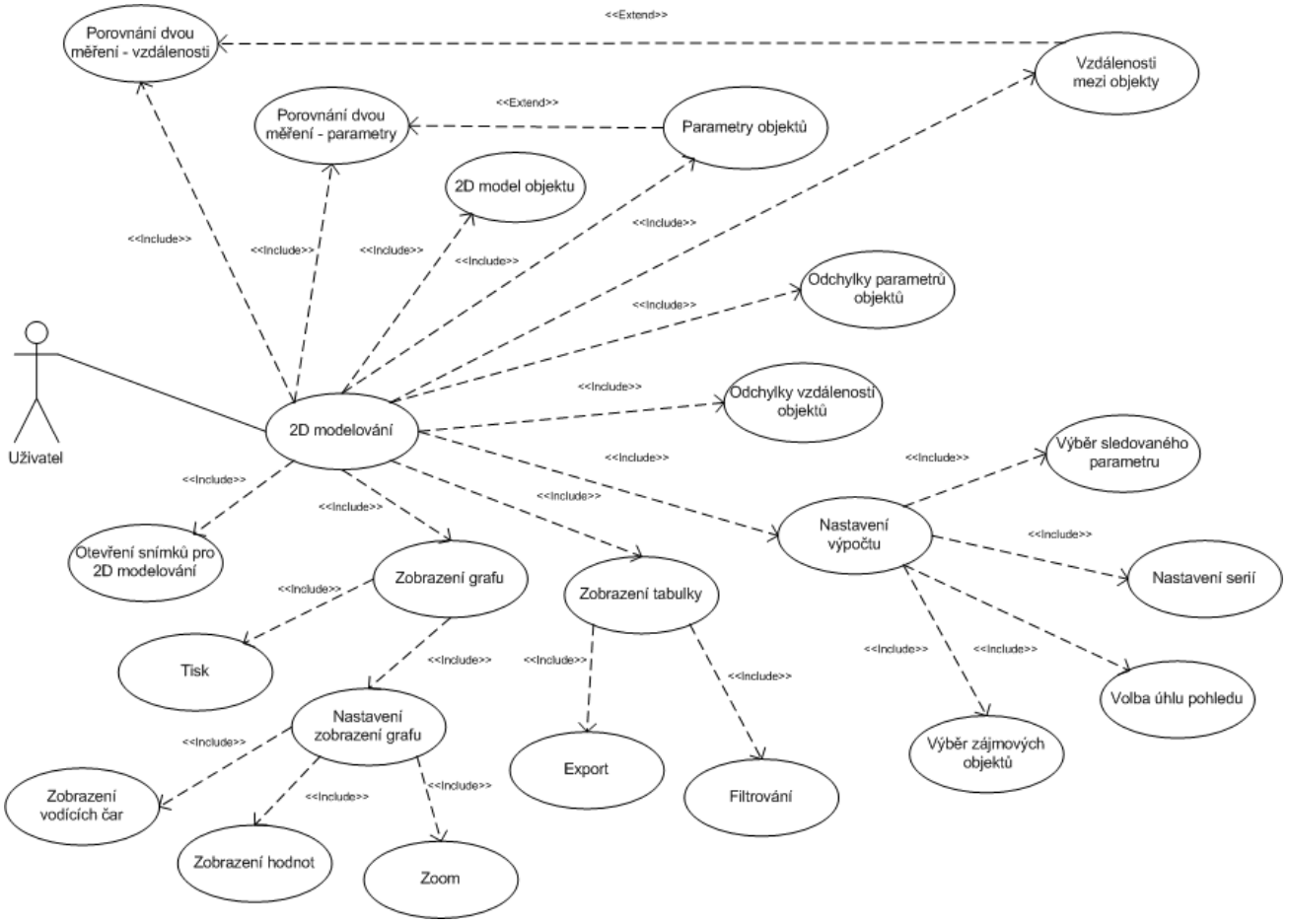
Na obrázku 8 je zobrazen diagram případů užití modulu pro 2D modelování. Diagram charakterizuje užívání systému účastníkem (tedy uživatelem) a vymezuje oblast modelovaného problému. Na základě funkcionality, která je popsána ve specifikaci, ilustruje jednotlivé procesy, které budou v modulu zahrnuty, a jejich asociace.

Celková práce s modulem je shrnuta v globálním případu užití *2D modelování*. Případ užití *Otevření snímků pro 2D modelování* popisuje postup při otevírání snímků pro 2D modelování a také přidávání druhé série anebo souboru s projektovanými hodnotami k již existujícímu modelování.

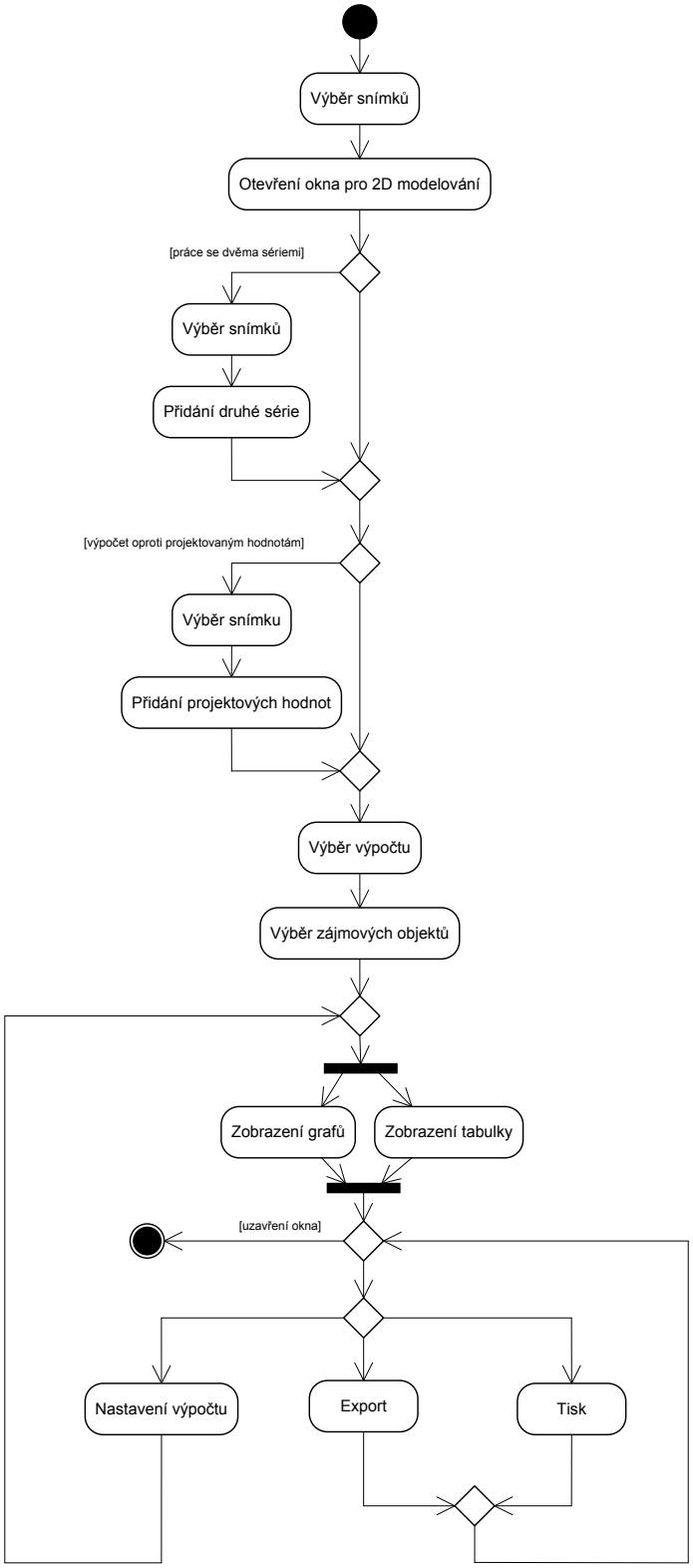
Zobrazení grafu a *Zobrazení tabulky* jsou případy užití, které specifikují oba dva způsoby zobrazení výstupních dat kalkulací. Každý způsob je velmi odlišný, co do možností nastavení anebo dílčích operací. Grafické zobrazení umožňuje vytisknout výsledky, použít funkci zoom anebo upravit vzhled grafů. Tyto operace popisují případy užití *Tisk* a *Nastavení zobrazení grafu*. Tabulka poskytuje podle specifikace požadavků uživateli možnost filtrace a exportu dat, což jsou operace popsané případy užití *Export* a *Filtrování*.

V podkapitole 4.3 jsme specifikovali celou škálu různých typů výpočtů. Každý z nich je v diagramu reprezentován svým vlastním případem užití. Jednotlivé možnosti nastavení vstupu pro tyto výpočty jsou vymezeny v případu užití *Nastavení výpočtu*.

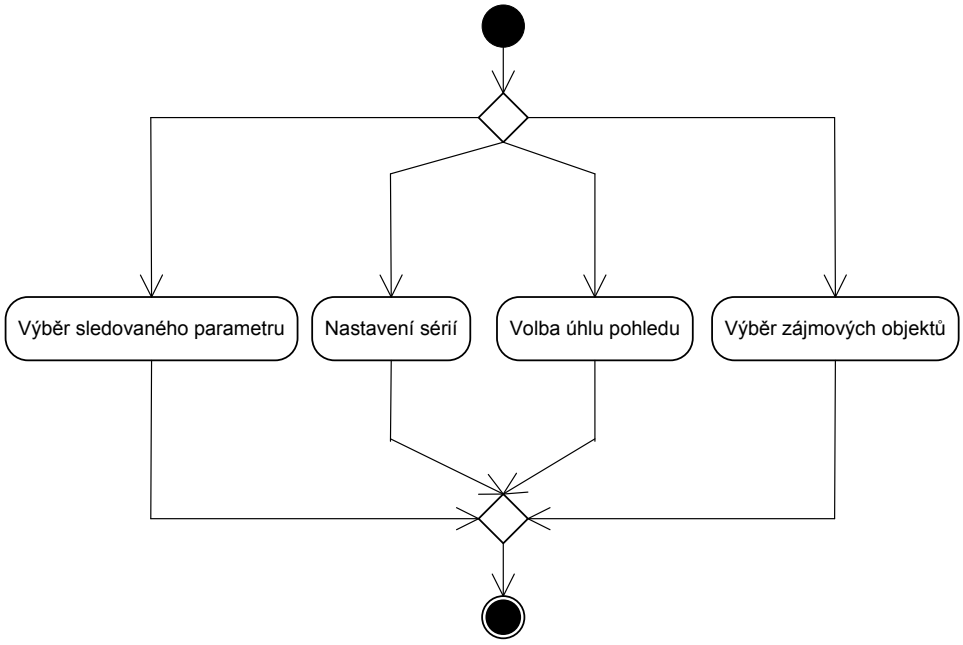
Pojďme se nyní zaměřit na to, jak by mělo vypadat chování modulu. Diagram aktivit na obrázku 9 zobrazuje dynamiku procesů modelovaného systému. Jak je patrné, dojede s prvním výběrem snímků k otevření okna pro 2D modelování. Poté modul umožní přidat druhou sérii anebo projektové hodnoty. Dále si vyžádá zadání typu výpočtu a výběr objektů, nad kterými bude počítáno. Následně zobrazí grafickou a tabulkovou reprezentaci výsledku a umožní uživateli s nimi dále pracovat. Při změně nastavení výpočtu (diagram aktivit na obrázku 10) dojede k novému zobrazení výsledků.



Obrázek 8 - Diagram případu užití navrhovaného modulu



Obrázek 9 - Diagram aktivit navrhovaného modulu



Obrázek 10 – Detailní diagram aktivit nastavení výpočtu

5.2 Analýza využitelnosti stávajícího API a návrh GUI

V následujícím textu navrhneme GUI nového modulu pro 2D modelování a vyhodnotíme, které části a objekty Fotom API zapojíme do návrhu vnitřní struktury.

5.2.1 Návrh GUI

V kapitole 3.2 jsme popsali vzhled a možnosti GUI aplikace Fotom NG. Nyní, když už známe požadavky na nový modul, je zapotřebí navrhnout, jaké ovládací prvky a možnosti využijeme a jak je začleníme do stávajícího GUI. Připomeňme, že jedním z požadavků je, aby bylo výsledné uživatelské rozhraní Fotomu NG integritní a využívalo stejných principů práce.

Z předcházející analýzy rovněž víme, že bude zapotřebí otevírat snímky. Tuto funkcionalitu obstarává Průzkumník Fotomu NG a je tedy nasnadě, že jej můžeme rozšířit tak, aby byl použitelný i pro navrhovaný modul. Toto rozšíření prakticky znamená přidání nových možností volby do kontextového menu. Budou to konkrétně tyto tři:

- Otevřít snímky pro 2D modelování
- Přidat jako druhou sérii
- Přidat jako projektované hodnoty

První možnost bude aktivní pouze tehdy, pokud bude v průzkumníku označeno vícero snímků. Výsledkem volby bude otevření okna modulu pro 2D modelování. Druhá možnost bude aktivní pouze tehdy, pokud bude zmíněné okno otevřeno a v Průzkumníku bude opět označeno několik snímků. Výsledkem operace bude zaregistrování označených snímků v právě aktivním okně modulu jakožto druhé série. Třetí možnost připadne v úvahu pouze, když bude otevřeno okno pro 2D modelování a v průzkumníku bude označen právě jeden snímek. Výsledkem provedení volby bude zaregistrování

snímku do právě aktivního modelování jakožto snímku, z něhož budou odečítány projektované hodnoty.

Zmíněné okno pro 2D modelování bude navrženo podle vzhledu stávajících oken pro editaci snímku. Bude se otevírat ve formě záložky v centrální části obrazovky a bude obsahovat panel pro rychlý přístup k nejčastěji používaným funkcionalitám. V hlavní části okna budou zobrazeny grafické výstupy výpočtů.

Výběr typu výpočtu bude realizován přes nové menu umístěné v panelu hlavního menu. Menu bude obsahovat položku pro každý výpočet.

Z analýzy také vyplývá, že je po uživateli požadováno, aby určil, nad kterými zájmovými objekty bude výpočet prováděn. Požadavky na tento úkon prakticky splňuje již implementovaný Manažer objektů. Bude tedy vhodné jej rozšířit tak, aby komunikoval s novým modulem.

Poslední části specifikace, jejíž reprezentaci v GUI jsme dosud nenavrhli, je tabulkové zobrazení výsledků. Při zvážení smyslu modulu je zřejmé, že nebude pro svou funkci využívat stávající komponenty Paleta a Vlastnosti. Je tedy vhodné toto místo využít pro zobrazení funkcionality, kterou potřebujeme, tedy konkrétně zmíněné tabulky. Tabulka tedy bude reprezentována novým oknem, které se bude defaultně zobrazovat po celé části levé strany aplikace.

5.2.2 Analýza Fotom API

Aplikační rozhraní Fotomu NG obsahuje několik struktur, které můžeme využít. Mezi ty nejdůležitější patří:

FtmObject

Tato třída v sobě nese veškeré informace o snímku a vytváří tak jednotný mechanismus přístupu ke snímku a jeho vlastnostem [1]. Součástí instance této třídy je vždy také tzv. *kontejner*, který v sobě sdružuje informace o zájmových objektech na snímku, lícovacích bodech, měřících jednotkách atd. Následující ukázka ilustruje načtení objektů snímku a hodnoty global Z:

```
int globalZ = ftm.getContainter().getMeasureDef().getGlobalZ();
ShapeTool[] tools = ftm.getContainter().getList();
```

ShapeTool a zájmové objekty

V ukázce byla zmíněna třída *ShapeTool*. Jedná se o abstraktní třídu odvozenou od obecné třídy *Tool*, která reprezentuje nástroje pro kreslení na plátno. *ShapeTool* se vymezuje na nástroje pro definování tvarů na snímku a proto prakticky zosobňuje námi využívané zájmové objekty. Potomky třídy *ShapeTool* jsou konkrétní zástupci objektů jako *PointTool*, *CrossTool*, *CircleTool* a *PolygonTool*. Tyto třídy v sobě zahrnují spoustu předpřipravené funkčnosti, díky které můžeme velmi snadno získat souřadnice zájmového bodu nebo obsah obrazce ve skutečných jednotkách:

```
CircleTool circle = (CircleTool) tool;
circle.init();
double xValue = circle.getRealPointOfInterest().getX();
double yValue = circle.getRealPointOfInterest().getY();
double areaValue = circle.getRealArea();
```

Poznamenejme ještě na okraj dvě skutečnosti – třída *ShapeTool* ani z ní odvození třídy neobsahují metody pro výpočet obvodu obrazce a proto bude třeba tuto funkcionalitu doimplementovat. Druhá poznámka se týká třídy *MeasureDef*, která je nepřímo zmíněna v prvním příkladu. Také ona je samozřejmě nástroj odvozený od abstraktní třídy *Tool*. Její konkrétní funkcí je

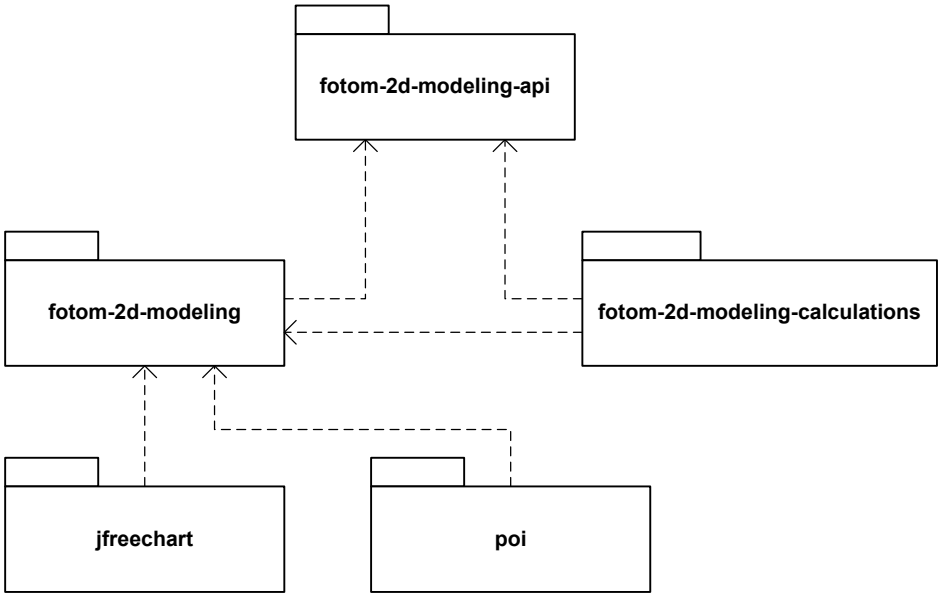
definice lícovacích bodů na snímku a krom metod pro získání global Z poskytuje také veškerou funkcionalitu pro přepočet mezi lokálními a globálními jednotkami na snímku.

5.3 Návrh vnitřní struktury a dílčích modulů

Díky vlastnostem platformy NetBeans můžeme celkovou funkcionalitu modulu pro 2D modelování rozvrhnout do několika dílčích modulů. Každý z nich bude realizovat určité oblasti problému a jejich správa a současný i budoucí vývoj budou ve výsledku velmi zjednodušeny. Podívejme se, jak budou jednotlivé dílčí moduly vypadat:

1. Modul pro API (fotom-2d-modeling-api)
2. Modul prezentační vrstvy (fotom-2d-modeling)
3. Modul pro výpočty (fotom-2d-modeling-calculations)
4. Modul s knihovnou JFreeChart (jfreechart)
5. Modul s knihovnou Apache POI (poi)

A nyní naznačme jejich vzájemné závislosti:



Obrázek 11 - Závislosti dílčích modulů

Na první pohled nemusí být zřejmé, proč je modul pro výpočty závislý na prezenční vrstvě. Je tomu proto, že výpočty definují grafické výstupy včetně barev jednotlivých křivek. Nastavení číselníku barev je plně v režii uživatele systému, který tak činí pomocí nástrojů prezentační vrstvy.

Dále si povšimněme, že je API zcela nezávislé na používané vykreslovací knihovně JFreeChart. Proč tomu tak je a jakým způsobem je toho dosaženo, je obsahem následujícího textu.

5.4 Modul pro API

Pro větší přehlednost si funkce API rozdělíme do několika podskupin. Každou z nich reprezentuje v modulu jeden balíček.

5.4.1 API pro 2D modelování

Obsahem této části je popis základních tříd realizující funkcionalitu modulu pro 2D modelování. Mezi ty nejdůležitější patří například:

FtmSeriesObject

Tato třída reprezentuje měření neboli sérii snímků. V podstatě se jedná o kolekci instancí *FtmObject*, která je rozšířena o další funkcionalitu. Třída má dva konstruktory – jeden pro vytvoření obecné série a další pro vytvoření související druhé série. Oba konstruktory kontrolují, zda předané snímky splňují kritéria pro vytvoření série. Tato kritéria jsme uvedli v kapitole 4.1.2. Pokud je některé z pravidel porušeno, konstruktor vyhodí patřičnou výjimku. Třídy výjimek jsou uloženy v balíku *exceptions* v rámci modulu. Hlavním rozdílem mezi oběma konstruktory je, že konstruktor pro druhou sérii vyžaduje jako vstupní parametr jinou (první) sérii a poté testuje, jestli tato odpovídá pravidlům pro druhou sérii vytvořenou vůči předané. Jak by měla validní druhá série vypadat, je opět popsáno v kapitole 4.1.2. Třída *FtmSeriesObject* si navíc na základě použitého konstruktoru pamatuje, zda byla série snímku vytvořena jako první nebo druhá. To nám usnadní identifikací sérii v lookupu.

FtmSeriesObject rovněž poskytuje metody pro načtení měřících jednotek nebo jednotek distance celé série. Dále umožňuje sérii setřídit podle hodnoty global Z, vrátit libovolný snímek série, přidávat nebo odebírat snímky atd.

Calculation

Abstraktní třída *Calculation* symbolizuje obecný výpočet a slouží jako vzorová třída pro všechny výpočty 2D modelování. Definuje několik abstraktních metod, které musí všichni její potomci implementovat.

První pomyslnou skupinu metod tvoří metody pro definici vstupu. Je zapotřebí, aby každý výpočet dokázal své nadřazené vrstvě sdělit, která data potřebuje na svém vstupu. Vrstva si může tyto informace zjistit voláním metod jako *requiresModeledParameter()*, *requiresTwoSeries()* atd.

Druhou skupinu tvoří metody pro předání výstupu výpočtu. Jsou to *getDataGraphs()* a *getDataTables()*. První vrací kolekci obecných grafů. Co je obecný graf a proč jej používáme je vysvětleno v kapitole 5.4.2. Metoda *getDataTables()* vrací strukturu Map, jejíž každý záznam obsahuje klíč ve formě textového řetězce a poté modifikovaný model JTable. Klíč reprezentuje název tabulky, o modifikovaném modelu JTable pojednává kapitola 5.4.3. Výsledné tabulky výpočtu lze také získat ve filtrované podobě, pokud se metodě *getDataTables()* předá instance třídy *CalculationFilter*, která udává nastavení filtru. Konkrétní výpočet umožňuje pouze některé typy filtrování. Které to přesně jsou, je možné zjistit voláním metody *getFilterPossibilities()*.

Za poslední skupinu metod třídy můžeme považovat interní metody *getFtmSeries()*, která získá potřebný počet sérií z globálního lookupu a *calculate()*, která provádí samotný matematický výpočet.

CalculationObject

Tato třída slouží pro vytvoření zástupného objektu, který v sobě bude uchovávat právě aktuální instanci výpočtu. V momentě, kdy je vytvořena komponenta s oknem pro 2D modelování, je také vytvořena instance *CalculationObject* a předán do lookupu komponenty. Díky vlastnostem kontextového lookupu a Fotom API, bude tento *CalculationObject* dostupný ze všech částí aplikace a díky řízení kontextu bude v globální lookupu vždy umístěn *CalculationObject* příslušející právě otevřenému oknu pro 2D modelování.

Během svého životního cyklu projde objekt několik stavy. Při vytvoření v sobě neobsahuje žádnou instanci třídy *Calculation* a je tedy prázdný. Pokud uživatel zvolí některý z výpočtů v hlavním

menu, je vytvořena příslušná třída a uložena do objektu *CalculationObject*. Jakákoliv komponenta systému může kdykoliv nahlédnout do globálního lookupu a použít nebo nastavit v něm uloženou kalkulaci.

Třída *CalculationObject* také umožňuje registrovat posluchače. Dojde-li tedy ke změně nastavení výpočtu anebo výpočtu samotného, budou všechny registrované komponenty upozorněny. Tímto způsobem lze snadno dosáhnout okamžité aktualizace grafů i tabulek.

Kromě kalkulace v sobě *CalculationObject* ukládá také informace o právě vybraném bodu. Nemáme nyní na mysli bod objektu, ale bod (hodnotu) grafu nebo tabulky. Tuto informaci reprezentuje třída *SelectedPointObject*. Díky výše popsanému čtení z lookupu tak může být synchronizováno označování bodu grafů a řádků tabulky, jak bylo požadováno v zadání.

GraphDisplayer

Tato abstraktní třída slouží k odvozování tříd, které budou zobrazovat grafická data. Konkrétně tato třída představuje právě jeden graf. Definuje základní metody pro práci s grafem jako vykreslení, překreslení a zoom. Abychom dosáhli toho, že je API nezávislé na zobrazovací knihovně, bude zapotřebí vytvořit potomka této třídy, který její metody přepíše v souladu s používanou knihovnou. V našem případě to bude třída *GraphDisplay*, kterou popíšeme v kapitole 5.5.1.

GraphDisplayerContainer

Protože výstupem výpočtu může být obecné množství grafů, musíme navrhnout kontejner, který je bude spravovat. Takovou strukturu definuje abstraktní třída *GraphDisplayerContainer*. Jedná se kolekci instancí třídy *GraphDisplayer*. Kontejner poskytuje metody, které zavolají operace vykreslení, překreslení a zoom pro všechny prvky v kolekci.

Kontejner by měl být rovněž zodpovědný za označování bodů v grafu. Protože je ale tato funkcionalita závislá na použité vykreslovací knihovně, je zapotřebí vytvořit a používat konkrétní instanci této třídy, která bude přizpůsobena na konkrétní knihovnu. V našem případě to bude třída *GraphDisplayContainer*, kterou popíšeme v kapitole 5.5.1.

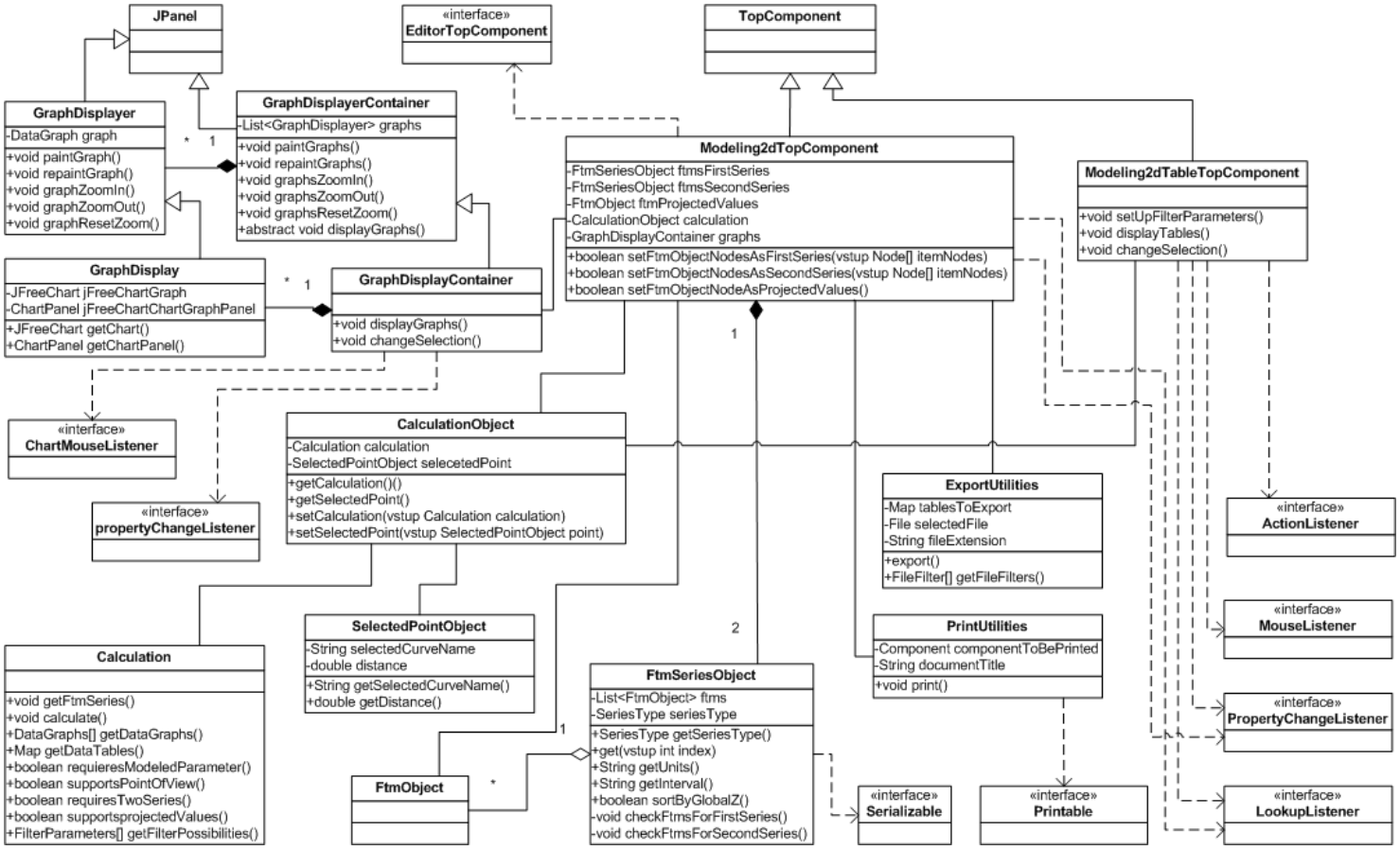
5.4.2 API pro definici obecného grafu

Již jsme zmiňovali, že je API modulu nezávislé na vykreslovací knihovně. V případě, že bude zapotřebí při budoucím vývoji nahradit tuto knihovnu novou verzí nebo případně úplně jiným vykreslovacím mechanismem, bude stačit napsat pouze dvě třídy, které s knihovnou bezprostředně pracují.

Na základě této myšlenky navrhneme jednoduché API pro obecný popis grafu. Výstupem každého výpočtu tak bude právě tato uniformní reprezentace a teprve ona bude dále zpracovávaná konkrétní vykreslovací knihovnou.

DataPoint

Tato abstraktní třída slouží jako vzor pro různé druhy datových bodů grafu. Třída určuje, že y-ová souřadnice bodu, která bude zastávat hodnotu parametru, bude vždy reálné číslo avšak x-ovou souřadnici, která bude znamenat číslo snímku anebo hodnotu distance, blíže nespecifikuje. Toto odpovídá požadavkům na budoucí rozšiřitelnost systému Fotom NG. V současné době používáme pouze číselnou variantu distance, ale s dalším vývojem jistě dojde k tomu, že distanc bude tvořit také časový údaj. V rámci API byly vytvořeny i dvě konkrétní třídy grafových bodů – třída *NumberDataPoint*, která je v aplikaci využívána a třída *DateDataPoint*, která je připravena pro rozšíření.



Obrázek 12 - Třídní diagram modulu pro 2D modelování

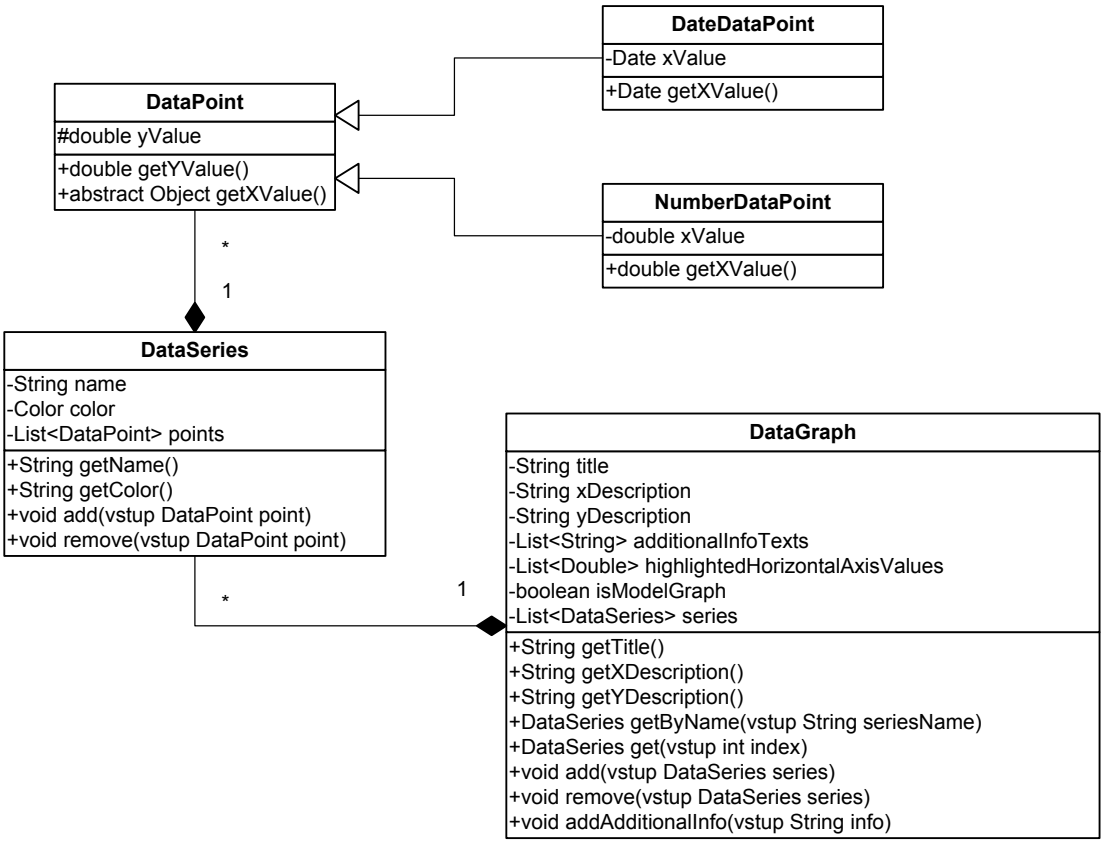
DataSeries

Třída *DataSeries* představuje jednu křivku grafu. Každá křivka má své jméno a vykreslovací barvu. Datová část je tvořena kolekcí bodů, tedy instancí třídy *DataPoint*.

DataGraph

Konečně tato třída reprezentuje samotný graf. Jejím základem je kolekce křivek, tedy instancí třídy *DataSeries*. Krom tohoto má každý graf svůj titulek a popisy x-ové a y-ové osy. Umožňuje vrátit křivky grafu podle indexu anebo podle jména, případně je vrátit ve formě pole.

V souladu s požadavky, které jsou na graf navrhovanou funkcionalitou kladeny, obsahuje třída také možnost přidat dodatečné informace do legendy grafu, vyznačit určitou y-ovou hodnotu, případně vypočíst maximum, minimum a průměr pro každou křivku grafu. Speciálně v sobě také uchovává informace, jestli se jedná o graf 2D modelu objektu. Takový graf se totiž bude vykreslovat jiným způsobem než běžný graf.



Obrázek 13 - Třídní diagram API pro definici obecného grafu

5.4.3 Rozšíření JTable modelu

Na základě analýzy bylo rozhodnuto, že se k vyobrazení tabulkového výstupu výpočtu bude využívat standardní javovský typ *JTable*. Komponenty tabulky tohoto typu ukládají svá data do *table modelu*. Existuje několik různých typů modelů, které jsou odvozeny od základní třídy *DefaultTableModel*. V podstatě by nám pro naše účely tento základní model vystačil, avšak s dalším

vývojem systému by se mohly vyskytnout okolnosti, které by si vyžádaly změnu modelu, a to by nezbytně vedlo k úpravám na několika různých místech aplikace.

To je důvod, proč byla vytvořena třída *Modeling2dTableModel*. Tato třída je přímým potomkem třídy *DefaultTableModel* a ve své současné verzi pouze drobně upravuje zobrazování záhlaví výsledných tabulek. Při dalším rozšíření aplikace bude nyní stačit pouze upravit třídu *Modeling2dTableModel*.

5.5 Modul prezentační vrstvy

V rámci modulu prezentační vrstvy jsou implementovány hlavně ty části aplikace, které budou komunikovat s uživatelem a prezentovat výstupy různých činností. Jedná se tedy o různé akce, třídy reprezentující komponenty GUI, třídy poskytující kontext nebo třídy realizující tisk a export.

5.5.1 Hlavní třídy a komponenty prezentační vrstvy

Modul prezentační vrstvy je stejně jako modul API přehledně rozdělen do několika balíků dle účelu a funkce. V následujícím textu se zaměříme na nejdůležitější třídy této oblasti.

GraphDisplay

Tato třída je konkrétní implementací abstraktní třídy *GraphDisplayer*. Dříve definované abstraktní metody přepisuje pomocí funkcionality zobrazovací knihovny JFreeChart. Třída se stará o vytvoření a zobrazení grafické reprezentace obecného grafu, který je výstupem některého z výpočtů. V prostředí JFreeChart realizuje všechna nastavení, která obecný graf má, jako je titulek grafu, zobrazení legendy, popis os nebo zvýraznění hodnoty.

GraphDisplayContainer

Stejně jako v předchozím případě je i tato třída konkrétní reprezentací abstraktního vzoru definovaného v API. Oproti svému předku je rozšířena o funkcionalitu reflektující nastavení zobrazení grafu jako je (de)aktivace popisků hodnot nebo vodících čar. Přidána byla také možnost zobrazit zprávu uživateli, pokud například neoznačil dostatečný počet zájmových objektů pro zvolený výpočet. Nejdůležitější částí je řízení označování hodnot grafů, které je prováděno pomocí dostupných prostředků knihovny JFreeChart a příslušný kód lze nalézt v metodě *changSelection()*.

Modeling2dTopComponent

Tato třída je ústřední komponentou modulu pro 2D modelování, neboť vytváří samotné okno modulu, které se zobrazí při otevření snímků do série. Instance je vytvářena akcí, která se zavolá při kliknutí na příslušnou položku kontextového menu v průzkumníku.

Třída poskytuje metody *setFtmObjectNodesAsFirstSeries*, *setFtmObjectNodesAsSecondSeries* a *setFtmObjectNodeAsProjectedValues*, které se pokusí vytvořit sérii snímků nebo přidat projektový soubor z označených nodů. Záměrně říkáme pokusí, protože může dojít k tomu, že snímky nebudou splňovat kritéria kladená na validní sérii. Metody v tomto případě oznámí uživateli, který požadavek snímky nesplňují a vrátí jako svou návratovou hodnotu *false*. Tato návratová hodnota putuje k akci, která řídí vytváření instancí a ta buď nové okno otevře anebo instanci zahodí.

V rámci grafické komponenty okna pro 2D modelování je také definován ovládací panel. Poskytuje uživateli následující možnosti:

- zoom grafů
- nastavení (aktivace/deaktivace) série

- nastavení sledovaného parametru
- nastavení zobrazení grafu (popisky dat, vodící čáry)
- tisk
- export
- nastavení úhlu pohledu

Třída eviduje jednotlivé akce panelu, a buď na ně přímo reaguje, nebo předává události dál, příslušným třídám či dialogům. O nastavení panelu se stará metoda *setupCalculation()*, která dle zvoleného výpočtu nastaví možnosti panelu.

Krom ovládacího panelu komponenta obsahuje instanci *GraphDisplayContainer*.

Modeling2dTableTopComponent

Druhou velmi důležitou komponentou pro prezentaci výsledků modelování je zobrazení tabulek s daty, které představuje třída *Modeling2dTableComponent*. Třída poslouchá změny objektu *CalculationObject* a reaguje na označení bodu na grafu nebo změnu výpočtu. Z výpočtu si pomocí volání metody *getDataTables()* přebere kolekci tabulkových modelů a jejich jmen a postupně je vykreslí pod sebou jako javovské komponenty *JTable*.

Pokud je v některé z tabulek označen řádek, třída sama vytváří instanci *SelectedPointObject* a umístí ji do aktuálního objektu *CalculationObject*.

Součástí komponenty je také formulář pro nastavení filtru. Po zadání hodnot a potvrzení formuláře vytvoří třída instanci třídy *CalculationFilter* a předá ho jako vstupní argument při volání *getDataTables()*. Tím získá z výpočtu pouze data filtrovaná dle nastaveného filtru.

5.5.2 Dialogy a nastavení modulu

Modul pro 2D modelování obsahuje právě dva dialogy pro interakci s uživatelem. V tomto případě nebereme v potaz dialogy oznámení ani dialogy operačního systému, které jsou systémem vyvolány.

Oba dva dialogy jsou umístěny v balíku *dialogs*. První z nich je dialog pro nastavení sérií a je vytvářen třídou *SeriesSettingsPanel*. Druhý umožňuje uživateli nastavit sledovaný parametr a je reprezentován třídou *ModeledParameterChooserPanel*.

Tyto třídy jsou velmi jednoduché a obsahují pouze minimum kódu. Jejich účelem je umožnit uživateli provést volbu a pak poskytnout veřejnou metodu, pomocí které si výsledek může přečíst mateřská komponenta *Modeling2dTopComponent*.

Už jsme mluvili o tom, že uživateli musí být umožněno nastavovat si vlastní číselník pro barvy křivek grafu. Toto nastavení je realizováno třídou *ColorsettingOptionsPanel*, která reprezentuje panel, který se přidá jako záložka do globálního dialogu nastavení aplikace. Třída je umístěna v balíku *options*.

5.5.3 Export a tisk

Třídy pro tisk grafů a export tabulek hodnot jsou umístěny v balíku *utilities*. Třída *PrintUtilities*, která obstarává tisk, je realizována pomocí standardních postupů jazyka Java.

Třída *ExportUtilities* slouží jako proxy třída, která v sobě eviduje všechny své dostupné potomky. Na základě typu exportu, který zjistí dle zvoleného filtru, zavolá příslušnou odvozenou třídu, která provede export. Třída *ExportUtilities* (a tím i její potomci) je sama rozšířením třídy *FileFilter*, takže poskytuje nastavení typů souborů pro standardní javovský dialog uložení.

O export do formátu CSV se stará třída *CSVExportUtilities*. Tento kód je založen čistě na skládání textových řetězců. Oproti tomu třídy *XLSExportUtilities* a *XLSXExportUtilities* využívají prostředky knihovny Apache POI.

5.6 Modul pro výpočty

Obsahem modulu pro výpočty jsou jednotlivé třídy kalkulací odvozené od abstraktní třídy *Calculation*. Každá z nich realizuje matematické výpočty podle svého účelu a vytváří výsledné definice obecných grafů a tabulkových modelů.

Každá třída výpočtu je umístěna v samostatném balíčku spolu s třídou akce, která zajišťuje vytvoření nové instance a uložení do aktuální instance třídy *CalculationObject*. V souboru layer.xml modulu jsou deklarovány položky pro menu 2D modelování v hlavním menu.

5.7 Moduly knihoven JFreeChart a Apache POI

Tyto moduly v sobě zastřešují veškeré třídy a knihovny potřebné pro běh JFreeChart a Apache POI. Byly vytvořeny jako obalové moduly pro knihovny (*Library Wrapper Module*) platformy NetBeans [9].

Standardní distribuce JFreeChart byla rozšířena o třídu *CircleDrawer*, která vykresluje kolečko na dané pozici v grafu a v modulu pro 2D modelování slouží k označování bodů grafu. Třída vychází z návrhu vývojáře JFreeChart pana Davida Gilberta [14].

6. Realizace a ověření funkčnosti

Implementace systému byla provedena na základě výsledků analýzy uvedené v předchozí kapitole. V následujícím textu se zaměříme na zvolené vývoje prostředí, výsledný vzhled systému a testování funkcionality.

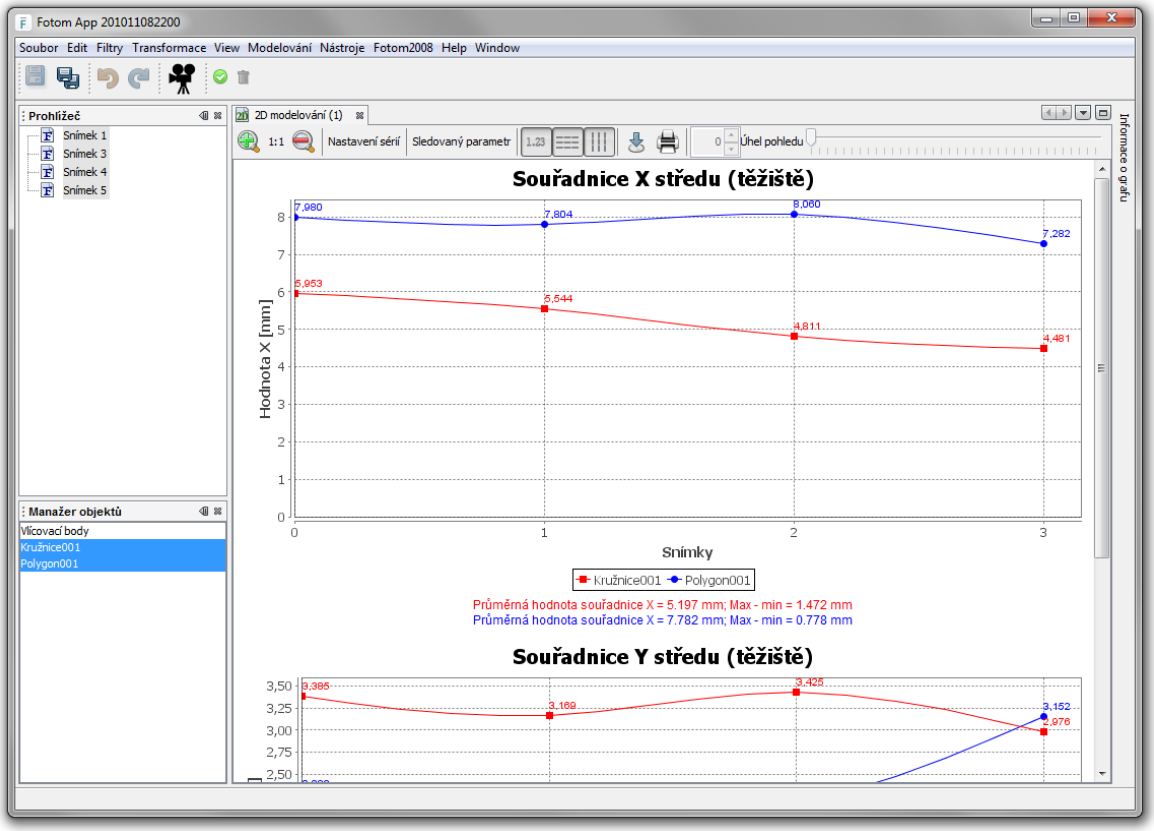
6.1 Vývojové prostředí a použité prostředky

Modul pro 2D modelování byl vyvinut ve vývojovém prostředí NetBeans IDE verze 6.9.1. Aplikace byla napsána v jazyce Java v distribuci JDK 1.6.0 update 23.

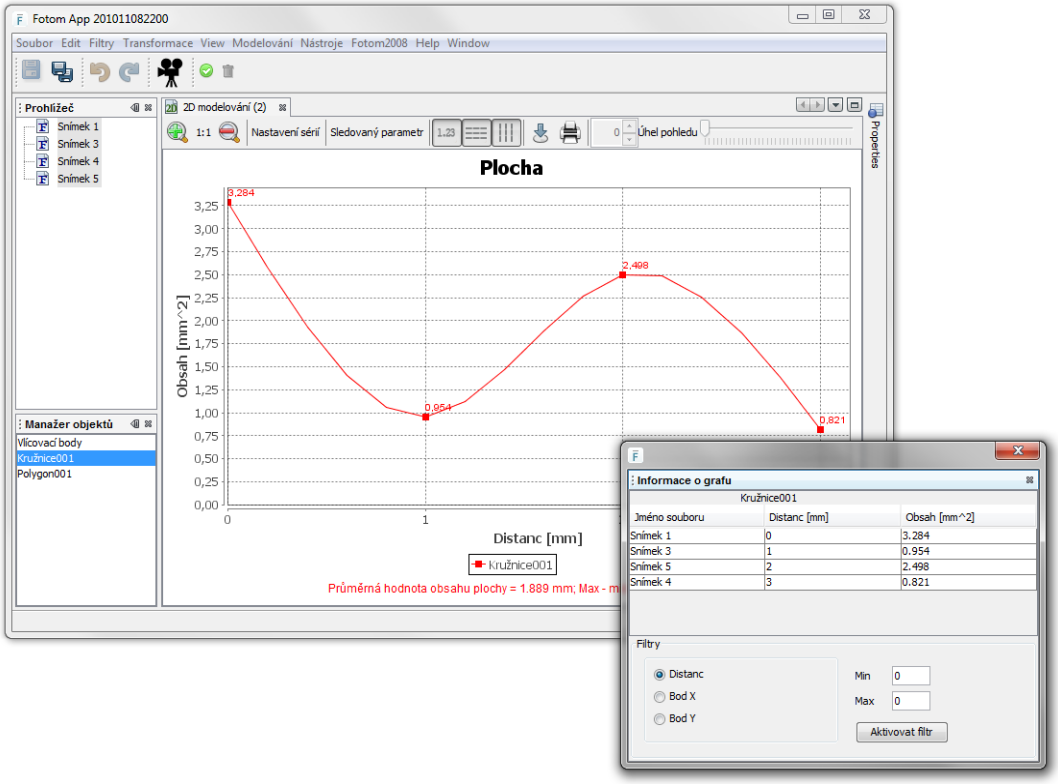
Pro realizaci grafických výstupů modul používá knihovnu JFreeChart ve verzi 1.0.13 a pro řízení exportu knihovnu Apache POI verze 3.7. Knihovny jsou součástí modulu a nemusí se samostatně instalovat. Modul však (stejně jako celý systém Fotom NG) vyžaduje nainstalovanou Java distribuci minimálně ve formě JRE.

6.2 Vzhled systému

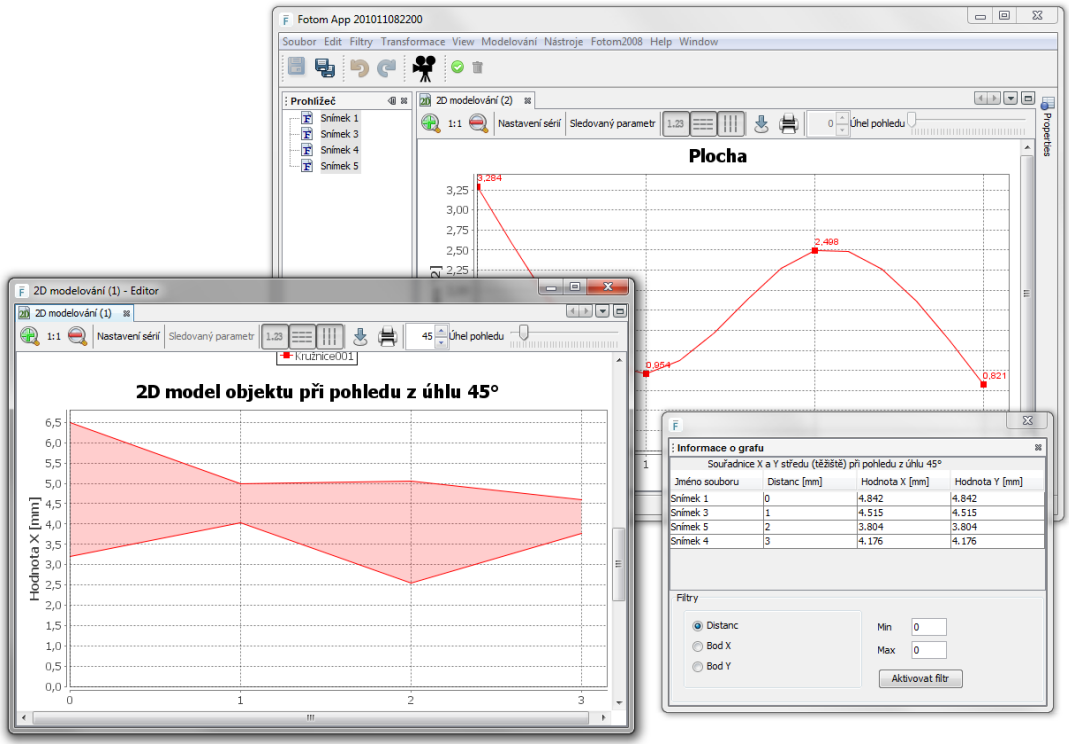
Uživatelské rozhraní bylo implementováno podle návrhu uvedeného v kapitole 5.2.1. Náhled na výslednou aplikaci je k dispozici na následujících obrázcích:



Obrázek 14 - Modul pro 2D modelování v rámci systému Fotom NG



Obrázek 15 - Sledování změn plochy objektu a tabulka hodnot v samostatném okně



Obrázek 16 - Souběžná práce s grafem, 2D modelem a tabulkou hodnot

6.3 Nutné zásahy do Fotom API

Již během analýzy jsme dospěli k názoru, že bude nezbytné na několika místech zasáhnout do původního API. Jednalo se především o rozšíření kontextového menu Průzkumníka a úpravu Manažera objektů. Tyto změny jsou detailněji popsány v příloze 5 na přiloženém CD. Žádná z těchto změn nemůže ohrozit stávající funkcionalitu nebo způsob využívání API.

6.4 Ověření funkčnosti

Modul 2D modelování pro systém Fotom NG byl implementován v plném rozsahu specifikovaných požadavků na základě výše uvedené analýzy.

Použité postupy, algoritmy a jednotky kódu byly napříč stádii vývoje průběžně testovány, aby se dosáhlo potřebné kvality po stránce verifikační i validační. Potenciálně nebezpečná místa kódu byla opatřena systémem výjimek a do GUI byly přidány dialogy pro sdělení uživateli.

Na základě specifikace požadavků byla vytvořena série testovacích případů, které se zaměřovaly na otestování hlavní funkcionality. Testovací případy s nejvyšší prioritou obsáhly tyto oblasti:

- operace se snímky a sériemi (vytvoření série, přidání druhé série)
- zobrazování grafických a tabulkových výstupů
- správnost a průběh výpočtů
- vzájemná komunikace mezi grafovými a tabulkovými výstupy
- uživatelské rozhraní

Testovací případy s druhotnou prioritou se soustředily na:

- filtrace tabulek
- nastavení zobrazení grafů
- export
- tisk

Všechny testovací případy byly spuštěny, dokončeny a proběhly úspěšně. Otestována byla také integrace do systému Fotom NG.

Navrhovaný modul je připraven na běžný provoz, avšak je zde stále riziko drobných chyb nebo nedostatků, které budou odstraňovány v průběhu nasazení systému. Jisté komplikace také mohou vyplynout z případných budoucích změn ve Fotom API.

7. Závěr

Úkolem této diplomové práce bylo specifikovat, navrhnout, implementovat a otestovat modul 2D modelování pro systém Fotom NG.

Od prvotní myšlenky k hotovému dílu vedla téměř rok a půl dlouhá cesta. Nejprve jsem se musel seznámit s vědním oborem fotogrammetrie, pochopit její principy a užívanou terminologii. Na základě těchto vědomostí jsem pak přistoupil ke studiu stávajících aplikací, jako byl druhý modul systému Fotom 2008 a samotný Fotom NG.

Mým cílem bylo obsáhnout Fotom 2 po stránce funkcionality – analyzovat výpočty, které provádí, prostudovat výstupy, které poskytují, prozkoumat uživatelské prostřední a způsob práce s programem. Díky těmto analýzám jsem pak mohl snadno určit slabá místa tohoto programu a specifikovat požadavky na nový modul tak, aby byla zachována funkcionalita a odstraněny nedostatky. V průběhu specifikace jsme rovněž na základě konzultací s vedoucím práce navrhli několik vylepšení a drobných úprav.

Nový modul pro 2D modelování měl být vyvinut tak, aby jej bylo možné integrovat do systému Fotom NG. Tento systém jsem tedy musel analyzovat jak po stránce funkcionality, tak po stránce implementace. Protože jsem do té doby neměl žádné zkušenosti s platformou NetBeans, začal jsem nejprve studiem tohoto softwarového prostředku. Poznání platformy mi pomohlo pochopit způsob fungování Fotomu NG a také jeho API.

Na základě provedené specifikace jsem ovšem usoudil, že platforma NetBeans a prostředky jazyka Java, ve kterém je systém Fotom NG napsán, nebudou pro implementaci navrhovaného modulu dostačující. Byl tady problém zobrazování grafů a exportu do různých formátů, které nešlo se dosavadními prostředky efektivně realizovat. Prozkoumal jsem tedy dostupná řešení a vybral jsem knihovny JFreeChart a Apache POI, u kterých jsem nastudoval způsob používání a dostupnou funkcionalitu.

Poté jsem přistoupil k analýze požadavků a k návrhu modulu. Díky studiu Fotomu NG jsem navrhl způsob práce s novým modulem a jeho GUI tak, aby korespondovalo s principy práce se stávajícím systémem Fotom NG. Tyto poznatky a znalost Fotom API mi pomohly vhodně navrhnout komponenty nového modulu a napsat jeho vlastní API. Využil jsem plně možností platformy NetBeans a rozdělil jsem řešený problém do dílčích modulů, kterým jsem určil vzájemné závislosti a způsob komunikace.

Na závěr jsem modul implementoval a integroval do systému Fotom NG. Otestoval jsem jak jeho funkcionalitu, tak důsledky integrace. Zároveň jsem definoval verze použitých softwarových prostředků a závislost aplikace na přeinstalovaných prostředcích operačního systému.

Veškeré vědomosti, které jsem získal, specifikace a provedené analýzy jsem shrnul v této práci. Hned v úvodu jsem popsal důvody, které vedly ke vzniku nového modulu pro 2D modelování. V další kapitole jsem se věnoval fotogrammetrii a principu 2D modelování. Obsahem další kapitoly se stala analýza stávajících systémů. V části se specifikací požadavků jsem nejprve shrnul všechny pojmy a termíny, které jsem se během studia tohoto problému naučil. Poté jsem definoval základní požadavky na navrhovaný modul a dále detailně popsal smysl a průběh jednotlivých výpočtů, které budou modulem uživateli nabízeny.

V analýze modulu jsem se věnoval rozdělení na dílčí moduly a vysvětlil jsem jejich účely a pravomoci. Zároveň jsem popsal nejdůležitější třídy modulů a definovaná API. Řešení jsem ilustroval pomocí vhodných diagramů.

Hotové dílo jsem otestoval dle testovacích případů sestavených na základě specifikace. Zaměřil jsem se na hlavní funkcionalitu a praktickou použitelnost. Na základě testů jsem rozhodl, že je systém

připravený pro nasazení, avšak připustil jsem možnost drobných nuancí, které se mohou během reálného užívání vyskytnout.

Modul pro 2D modelování byl navržen jako snadno upravitelná a rozšiřitelná komponenta systému Fotom NG. Pokud se tedy vyskytnou nové požadavky nebo požadavky na změny ze strany uživatelů systému, je modul otevřen pro další rozšíření a vylepšení. K tomuto by mohla přispět jak uživatelská, tak programátorská dokumentace, kterou jsem vypracoval a přiložil k této práci.

Díky novému modulu se Fotom NG stává zcela unikátním nástrojem pro zpracování snímků. Dobrým příklad je obrázek 16 v předcházející kapitole, který názorně ilustruje, jak snadno může uživatel systému sledovat měnící se parametr zájmového objektu (v tomto případě se jedná o plochu objektu), zároveň zobrazit korespondující 2D model a navíc také přesné hodnoty v tabulce. Fotom NG tak uživateli nabízí jedinečný a ucelený pohled na modelovanou oblast a velmi mu tak zjednodušuje formulování závěrů a odečítání výsledků. Žádný jiný současný systém pro 2D modelování či analýzu snímků takovou funkcionalitu nenabízí.

Literatura

[1] KRAHULEC, Lukáš. *Počítačové zpracování fotografie*. Ostrava, 2009. 62 s. Diplomová práce. Vysoká škola Báňská - Technická univerzita Ostrava

[2] KRÁL, Jan. *Počítačové zpracování fotografie*. Ostrava, 2010. 39 s. Diplomová práce. Vysoká škola Báňská - Technická univerzita Ostrava

[3] PAVELKA, Karel. *Fotogrammetrie*. 1. vydání. Plzeň : Západočeská univerzita v Plzni, 2003. 247 s. ISBN 80-7082-972-9

[4] MARŠÍK, Zbyněk. *Fotogrammetrie : Základy letecké fotogrammetrie*. 2. nezměněné vydání. Praha : Nakladatelství technické literatury, n. p., 1982. 148 s.

[5] BÖHM, Jozef. *Fotogrammetrie* [online]. 2002 [citováno 29. ledna 2011] Dostupné z WWW: <http://igdm.vsb.cz/igdm/materialy/Fotogrammetrie.pdf>

[6] Atherosclerosis. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 14 September 2002, last modified on 25 January 2011 [cit. 2011-02-01]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Atherosclerosis>

[7] ŽÍDEK, Rostislav. *2D modelování z fotografie na počítači*. Ostrava, 2008. 38 s. Bakalářská práce. Vysoká škola Báňská - Technická univerzita Ostrava

[8] KOŠTUŘÍK, Martin. *Počítačové zpracování fotografie*. Ostrava, 2000. 63 s. Diplomová práce. Vysoká škola Báňská - Technická univerzita Ostrava

[9] BÖCK, Heiko. *Platforma NetBeans : Podrobný průvodce programátora*. Vydání první. Brno : Computer Press, a.s., 2010. 320 s. ISBN 978-80-251-3116-9

[10] ZAKHOUR, Sharon, et al. *Java 6 : Výukový kurz*. Vydání první. Brno : Computer Press, a.s., 2007. 534 s. ISBN 978-80-251-1575-6

[11] *JFreeChart* [online]. 2005, 2009 [cit. 2011-04-22]. JFreeChart. Dostupné z WWW: <http://www.jfree.org/jfreechart/>

[12] *JFreeChart Class Library* [online]. 2000, 2009 [cit. 2011-04-22]. Overview. Dostupné z WWW: <http://www.jfree.org/jfreechart/api/javadoc/index.html>

[13] GILBERT, David. *The JFreeChart Class Library : Developer Guide*. Ostrava : Object Refinery Limited, 2007. 643 s.

[14] *Black Duck Koders.com* [online]. 2003 [cit. 2011-04-24]. Koders Code Search: CircleDrawer.java - Java - LGPL. Dostupné z WWW: <http://www.koders.com/java/fidC4548D03325DCA0DD9D64948EB27D41D8B7E9DD0.aspx>

[15] *The Apache POI Project* [online]. 2002, 2010 [cit. 2011-04-25]. Apache POI - the Java API for Microsoft Documents. Dostupné z WWW: <http://poi.apache.org/>

[16] *IETF Tools* [online]. October 2005 [cit. 2011-04-25]. Common Format and MIME Type for Comma-Separated Values (CSV) Files. Dostupné z WWW: <http://tools.ietf.org/html/rfc4180>

[17] *NetBeans* [online]. 2011 [cit. 2011-04-26]. NetBeans Platform. Dostupné z WWW: <http://netbeans.org/features/platform/index.html>

Přílohy

Přílohy a dodatky k této diplomové práci jsou uloženy na přiloženém CD. Jedná se o:

1. Zdrojové kódy modulu pro 2D modelování
2. Distribuce systému Fotom NG s integrovaným modulem pro 2D modelování
3. Uživatelská příručka modulu pro 2D modelování
4. Programátorská příručka modulu pro 2D modelování ve formě Javadoc
5. Dokumentace nutných změn v původním Fotom API